

TURINGMASCHINEN MIT ASSOZIATIVEM SPEICHERZUGRIFF

von
Reinhold Weicker

Erlangen 1975



Digitized by the Internet Archive
in 2026

TURINGMASCHINEN MIT ASSOZIATIVEM SPEICHERZUGRIFF

Dem Fachbereich Ingenieurwissenschaften
der Universität Erlangen-Nürnberg
zur Erlangung des Grades
DOKTOR - INGENIEUR

vorgelegt
von
Reinhold Weicker

Erlangen 1975



Als Dissertation genehmigt vom
Fachbereich Ingenieurwissenschaften
der Universität Erlangen-Nürnberg

Tag der Einreichung: 16.5.1975

Tag der Promotion: 11.7.1975

Dekan: Prof. Dr. H.J. Schneider

Berichterstatter: Prof. Dr. H.J. Schneider

Prof. Dr. H. Müller

Prof. Dr. W. Brauer

Zusammenfassung

Es wird das Modell der Turingmaschine mit assoziativem Speicherzugriff (TAS) definiert und im Hinblick auf den dadurch bestimmten Komplexitätsbegriff untersucht. Diese Maschine arbeitet wie eine Turingmaschine, kann jedoch zusätzlich Marken auf Bandfelder setzen und auf so markierte Felder später in einem Schritt springen. Die Zusammenhänge zwischen der Anzahl dieser Marken und der Zeitkomplexität werden untersucht sowie Hierarchiesätze für Zeitkomplexitätsklassen bewiesen. Die Komplexitätsklassen der TAS und der Random-Access-Maschine (RAM) unterscheiden sich nur geringfügig, es gilt

$$C_{RAM}^1(T(n)) \leq C_{TAS}^{1,\infty}(T(n)^{1+\varepsilon}) \quad \text{für jedes } \varepsilon > 0,$$

$$C_{TAS}^{\infty,\infty}(T(n)) \leq C_{RAM}^1(T(n) \cdot \log L(n)).$$

Schließlich wird eine Random-Access-Maschine mit assoziativem Speicherzugriff (RAS) definiert und gezeigt, daß dieser assoziative Zugriff höchstens einen logarithmischen Gewinn an Rechenzeit bringt, es gilt

$$C_{RAS}^1(T(n)) \leq C_{RAM}^1(T(n) \cdot (\log T(n))^2).$$

Abstract

The model of a Turing machine with associative memory access (TAS) is defined and the complexity measure given by this model is investigated. This machine acts like a Turing machine, but it has the additional ability to set markers on the tape cells and to jump to marked cells in one step. Theorems about the relations between complexity classes and the number of markers as well as hierarchy theorems for complexity classes are proved. The complexity classes of the TAS and the Random access machine (RAM) are fairly close together, we have

$$C_{RAM}^1(T(n)) \leq C_{TAS}^{1,\infty}(T(n)^{1+\varepsilon}) \quad \text{for every } \varepsilon > 0,$$

$$C_{TAS}^{\infty,\infty}(T(n)) \leq C_{RAM}^1(T(n) \cdot \log L(n)).$$

Finally a Random access machine with associative memory access (RAS) is defined, it is shown that this associative access yields not more than logarithmic speed-up, we have

$$C_{RAS}^1(T(n)) \leq C_{RAM}^1(T(n) \cdot (\log T(n))^2).$$

I n h a l t

1. Einleitung	1
2. Definition der TAS und elementare Eigenschaften	8
3. Relative Leistungsfähigkeit von TAS, 1TAS und TM (On-Line-Modelle)	20
4. Markenzahl und Komplexität bei der TAS	41
5. Hierarchiesätze für Komplexitätsklassen der TAS	57
6. Beziehungen zwischen TAS und RAM	69
7. Random-Access-Maschinen mit assoziativem Speicherzugriff	92
Literatur	116

1. Einleitung

In der Komplexitätstheorie geht es um die Frage, wie schwierig Funktionen zu berechnen oder Algorithmen auszuführen sind, wie "komplex" sie sind. Als wichtiger Maßstab für die Komplexität gilt dabei der Zeitaufwand, d.h. die Frage, welche Anzahl von Schritten - in Abhängigkeit von den Eingabedaten - ein Algorithmus benötigt. Es ist klar, daß hierfür die Definition, was als Schritt anzusehen ist, eine wesentliche Rolle spielt. Da aus der Theorie der Berechenbarkeit bekannt ist, daß alles, was überhaupt als berechenbar angesehen werden kann, durch eine Turingmaschine berechenbar ist (Church-Turing'sche These), und da die Turingmaschine das bekannteste allgemeine Modell in der theoretischen Informatik darstellt, lag es nahe, für Komplexitätsbetrachtungen als Elementaroperation einen Schritt einer (mehrbändigen) Turingmaschine (TM) anzusehen; so geschah es in dem grundlegenden Aufsatz von Hartmanis und Stearns [HS 65], der diesem Gebiet den Namen gegeben hat. Obwohl in den letzten Jahren als Automatenmodell für Komplexitätsbetrachtungen die Random-Access-Maschine (RAM) mehr in den Vordergrund trat, bleibt doch die TM ein wichtiges Automatenmodell auch für Komplexitätsbetrachtungen, und es werden auch immer wieder Beziehungen zwischen den Problemen für jedes der beiden Modelle gefunden (z.B. [HS 74]).

In dieser Arbeit wird vor allem das Modell der Turingmaschine mit assoziativem Speicherzugriff (TAS) betrachtet.

Die Betrachtung eines solchen Maschinenmodells kann unter mehreren Aspekten von Interesse sein: Zum einen ist der wichtigste Nachteil einer Turingmaschine bei Komplexitätsbetrachtungen, daß eine TM, die einen bestimmten Algorithmus ausführt, einen großen Teil ihrer Schritte gar nicht zur eigentlichen Berechnung, sondern nur zum Verschieben ihres Lesekopfes auf dem Band benötigt, um an Daten heranzukommen, die sie im Verlauf ihrer Berechnung auf dem Band abgelegt hat. Solche Schritte, die nach einer intuitiven Vorstellung gar nichts mit der eigentlichen Berechnung zu tun haben, können bei der TAS durch die Möglichkeit, mit Hilfe von Marken größere Bandabschnitte zu überspringen, eingespart werden. Die TAS erscheint daher als ein Modell, das gerade solche Schritte, die im Sinne einer Komplexitätsbetrachtung irrelevant sind, eliminiert, das im übrigen aber das Konzept der Turingmaschine beibehält. Zum anderen stellt die TAS, wie schon der Name andeutet, ein Modell für eine Speicherung mit assoziativer Zugriffsmöglichkeit dar: Auf welches Speicherelement (Feld des Bandes) jeweils als nächstes zugegriffen wird, hängt von dem dort gespeicherten Datum ab. Im Sinne einer bei assoziativer Speicherung häufig benutzten Unterscheidung zwischen "Schlüsselworten" und "Information" stellen im Bandfeld der TAS die dort gesetzten Marken das Schlüsselwort und das Bandsymbol die Information dar.

Vor einem Mißverständnis, das durch die Verwendung des Wortes "assoziativ" entstehen könnte, muß gewarnt werden: Es handelt sich bei dem hier beschriebenen Maschinenmodell

zwar um eine Speicherungsform, die einen vom Inhalt her gesteuerten Zugriff im Sinne eines "associative memory" ermöglicht - und insofern ist das Wort "assoziativ" gerechtfertigt -, es geht aber nicht um eine assoziative Verarbeitung in dem Sinne, daß in mehreren Speicherzellen, die den gleichen Inhalt haben, gleichzeitig Operationen vorgenommen werden ("associative processing"). Bei der TAS wird immer nur auf ein Bandfeld assoziativ zugegriffen, von dort aus kann evtl. ein weiterer Sprung auf das nächste markierte Bandfeld erfolgen usw. Auch unter einem anderen Aspekt unterscheidet sich die TAS von den Modellen, die in der technisch orientierten Literatur über assoziative Speicher üblich sind: Wenn mehrere Bandfelder gleich markiert sind, dann erfolgt der assoziative Zugriff nicht immer auf das erste von ihnen, sondern auf das zur aktuellen Stellung des Lesekopfes am nächsten liegende Feld. Im Sinne einer Unterscheidung zwischen Notierungs- und Ordnungsprinzip bei der Speicherung von Information, wie sie etwa von E.Jessen in [Je 65] dargestellt wird, ist für die TAS also auch beim assoziativen Zugriff ein Ordnungsprinzip (Reihenfolge der Felder auf dem Band) maßgeblich beteiligt. In Abschnitt 3 wird gezeigt, daß es demgegenüber eine starke Einschränkung bedeutet, wenn der assoziative Zugriff immer auf das erste markierte Bandfeld erfolgt.

Zum Schluß der Arbeit wird die Idee des assoziativen Speicherzugriffs auch auf das Modell der Random-Access-Maschine (RAM) übertragen und eine RAS (Random-Access-Maschine mit assoziativem Speicherzugriff) definiert. Die Aussagen über dieses Modell könnten von einem praktischen Ge-

sichtspunkt her eher von Interesse sein als Aussagen über die TAS, die wesentlichen Beweisideen sind jedoch ganz ähnlich.

Das Modell der TAS wurde unter dem Namen "Tabulator-Turingmaschine" in [We 71] eingeführt, nachdem in der Diplomarbeit des Autors [We 70] verschiedene Erkennungsalgorithmen für kontextfreie Sprachen im Hinblick auf ihre Komplexität bei der Realisierung durch eine solche Maschine untersucht worden waren. Einige weitere Ergebnisse über die TAS enthält [WM 73], die meisten der hier wiedergegebenen Ergebnisse sind in einer verkürzten Form (die Beweise sind meist nur angedeutet) auch in [We 74] enthalten.

Die Möglichkeit, eine Turingmaschine mit dieser Art assoziativen Speicherzugriffs zu definieren, wurde auch von J.Hartmanis in [Ha 71a] erwähnt; er erläutert dort die Schwierigkeit, Resultate über die Komplexitätsklassen dieses Maschinentyps zu gewinnen. Auch S.Greibach fragte in [Gr 74] nach der Existenz einer, wie sie es nannte, "'Jump' Hierarchy" von Komplexitätsklassen für Turingmaschinen mit einer Sprungmöglichkeit. In Abschnitt 4 und 5 wird gezeigt, wie die von Hartmanis genannte Schwierigkeit durch eine geeignete Strukturierung der gespeicherten Information überwunden werden kann. Ein ähnliches Modell einer Turingmaschine mit assoziativem Speicherzugriff, jedoch mit zwei wesentlichen Einschränkungen, wurde von M.J.Fischer und A.L.Rosenberg betrachtet [FR 68].

In dieser Arbeit werden zunächst in Abschnitt 2 das Modell der TAS und einer in der Sprungmöglichkeit einge-

schränkten Variante, der 1TAS formal definiert. Diese Definition dient als formale Grundlage für die folgenden Beweise, bei denen allerdings der besseren Anschaulichkeit wegen das Verhalten einer TAS meist eher informal beschrieben wird. In Abschnitt 3 werden für den Fall der On-Line-Spracherkennung die Modelle der TAS, der 1TAS und der normalen TM miteinander verglichen. Es zeigt sich insbesondere, daß die Einschränkung bei der 1TAS, daß Sprünge nur vom Bandanfang aus zugelassen sind, zu einem merklich größeren Zeitaufwand führen kann. In Abschnitt 4 wird der Einfluß der Anzahl der Marken auf die Komplexität untersucht, es wird gezeigt, daß für jedes k eine TAS mit einer beliebigen, aber endlichen Anzahl von Marken, Bandbeschränkung $L(n)$ und Zeitbeschränkung $T(n)$ durch eine TAS mit nur k Marken und Zeitbeschränkung $C.k.\sqrt[k]{L(n)}.T(n)$ simuliert werden kann. Dieses Ergebnis wird in Abschnitt 5 benutzt, um die Existenz einer Hierarchie von Komplexitätsklassen für die TAS zu zeigen, die nur von der Zeitbeschränkung, nicht aber von der Anzahl der Marken abhängt. Hierarchiesätze, die für Komplexitätsklassen mit einer festen Anzahl von Marken gelten, werden ebenfalls bewiesen. Abschnitt 6 zeigt, daß es eine relativ enge Verbindung zwischen den Komplexitätsklassen der TAS und der RAM gibt. Es wird bewiesen, daß für jedes $\epsilon > 0$ eine RAM durch eine TAS mit Zeitbeschränkung $\frac{C}{\epsilon}.T(n)^{1+\epsilon}$ simuliert werden kann, andererseits gibt es zu jeder TAS eine äquivalente RAM mit Zeitbeschränkung $C.T(n).\log L(n)$. Schließlich wird in Abschnitt 7 das Konzept des assoziativen Speicherzugriffs auch auf die

RAM übertragen und eine RAS (Random-Access-Maschine mit assoziativem Speicherzugriff) definiert. Es zeigt sich, daß der dadurch evtl. erzielte Zeitgewinn nicht allzu groß ist; jede RAS läßt sich durch eine RAM mit Zeitbeschränkung $C.T(n).(\log T(n))^2$ simulieren. Eine Simulation einer RAS durch eine RAM mit Zeitbeschränkung $C.T(n).(\log T(n))^3$ ist auch dann noch möglich, wenn ein assoziativer Zugriff nicht vom gesamten Inhalt eines Registers, sondern vom Inhalt einer Anzahl führender Binärstellen des Registers abhängt ("Präfix-RAS").

Diese Einleitung soll nicht ohne einen besonderen Dank an Herrn Prof.Dr. Monien abgeschlossen werden, dem ich eine Reihe von Gesprächen und mehrere Hinweise zu der Arbeit verdanke. Mein Dank gilt auch Herrn Prof.Dr. Brauer für seine Förderung und Anteilnahme, außerdem Herrn G.Fleischhauer, der ebenfalls - im Rahmen einer Diplomarbeit - die Komplexität von verschiedenen Automatenmodellen untersuchte.

2. Definition der TAS und elementare Eigenschaften

Eine Turingmaschine mit assoziativem Speicherzugriff (TAS) läßt sich formal als 7-Tupel $\tilde{T} = (A, X, M, Z, Z_F, z_0, \delta)$ beschreiben mit

A Eingabealphabet

X Bandalphabet

M endliche Menge von Marken $m_1, \dots, m_k$

$Z \subseteq Z_S \times Z_M$ endliche Menge von Zuständen (Z_S "Speicherkomponenten", Z_M "Markenkomponenten") mit

$$Z_M = \{0, 1, \dots, |M|\}, \text{ also } |Z_M| = |M| + 1$$

$Z_F \subseteq Z$ Endzustände

$z_0 \in Z$ Anfangszustand

$$\delta: Z \times A \times X \longrightarrow Z \times X \times \{0, 1\} \times \{-1, 0, 1\} \times \{-1, 0, 1\}^{|M|}$$

Überführungstransformation, die auf einer (echten) Teilmenge von $Z \times A \times X$ erklärt und dort eindeutig ist ($\tilde{T}$ soll also immer eine deterministische Maschine sein).

O.B.d.A. enthalte das Bandalphabet immer die Zeichen b (Leerzeichen), \$ und ¢ (linkes bzw. rechtes Begrenzungszeichen), $\neg \notin A$ bezeichne das Ende der Eingabe.

Die Arbeitsweise der Maschine ergibt sich, indem die Abbildung

$$\delta(z, a, x) = (z', x', d_1, d_2, j_1, \dots, j_{|M|})$$

folgendermaßen als Beschreibung einer Maschinenoperation gedeutet wird:

Die Maschine besitzt ein Eingabeband mit einem Lesekopf, der nur in einer Richtung bewegt werden kann, und ein

Arbeitsband mit einem Lese- und Schreibkopf. Das Arbeitsband ist in zwei Spuren unterteilt, eine für den eigentlichen Bandinhalt, die Symbole aus X (Symbolspur), und eine für die Marken aus M (Markenspur). Dabei kann ein Feld mehrere verschiedene Marken gleichzeitig, maximal also $k = |M|$ Marken enthalten. z sei der augenblickliche Zustand der Maschine, a das Symbol unter dem Lesekopf des Eingabebandes, x das Symbol unter dem Lesekopf des Arbeitsbandes. Die Abbildung $f(z, a, x) = (z', x', d_1, d_2, j_1, \dots, j_k)$ beschreibt dann den folgenden Schritt:

Zustandsübergang: Die Maschine geht vom Zustand z in den Zustand z' über.

Bewegung des Lesekopfes (Eingabeband): Der Lesekopf auf dem Eingabeband rückt bei $d_1 = 1$ ein Feld weiter, bei $d_1 = 0$ bleibt er für diesen Schritt stehen.

Schreiben: Auf dem Arbeitsband wird in der Symbolspur das Symbol x durch das Symbol x' ersetzt. In der Markenspur enthalte das Feld unter dem Lesekopf einige der Marken $m_1, \dots, m_k$. Für jede Marke m_i ($1 \leq i \leq k$) gibt dann j_i an, ob m_i gelöscht wird (im Fall $j_i = -1$) oder gesetzt wird ($j_i = 1$). Im Fall $j_i = 0$ ändert sich in Bezug auf die Marke m_i nichts, sie bleibt im Feld unter dem Lesekopf genau dann stehen, wenn sie sich vorher dort befand.

Bewegung des Lesekopfes (Arbeitsband): Die Fortbewegung des Lesekopfes auf dem Arbeitsband hängt davon ab, ob für die zweite Komponente ("Markenkomponente") z_2 des Zustands $z = (z_1, z_2)$ $z_2 = 0$ oder $z_2 > 0$ gilt (Die "Speicherkomponente" z_1 dient wie üblich zur Realisierung eines endlichen Gedächtnisses). Im Fall $z_2 = 0$ rückt der Lesekopf, wie üblich,

ein Feld nach links, nach rechts, oder er bleibt stehen, je nachdem ob $d_2 = -1, 1$ oder 0 ist. Andernfalls sei $z_2 = 1$, $1 \leq i \leq k$. Der Lesekopf rückt dann in diesem einen Schritt der TAS so weit nach links (im Fall $d_2 = -1$) bzw. nach rechts (im Fall $d_2 = 1$), bis sich auf der Markenspur des Bandes die Marke m_1 unter dem Lesekopf befindet. Eine Marke m_1 kann auf dem Band mehrfach vorkommen, ein Sprung erfolgt jeweils auf die nächste Marke m_1 in der durch d_2 bestimmten Richtung. Die Begrenzungszeichen an beiden Enden des jeweils beschriebenen Bandabschnittes werden immer mit allen Marken aus M markiert, so daß ein Sprung spätestens dort endet und nie "ins Leere" geht.

Schließlich sei δ immer so definiert, daß im Fall, daß $z_2 > 0$ ist, auch $d_2 \neq 0$ gilt. Wie üblich, soll δ außerdem der Einschränkung unterliegen, daß ein Begrenzungszeichen vom Lesekopf höchstens weitergeschoben, nicht aber überschritten wird. Es soll also für $z \in Z$ entweder

$\delta(z, a, \$) = (., \$, ., 1, j_1, \dots, j_k)$ mit $j_1 = \dots = j_k = 1$ gelten oder

$$\delta(z, a, \$) = (z', x', ., -1, \dots)$$

und $\delta(z', a', b) = (z'', \$, ., j_1, \dots, j_k)$ mit $j_1 = \dots = j_k = 1$ für alle $a, a' \in A$; analoges gelte für das rechte Begrenzungszeichen ¢ . Dabei wird angenommen, daß die Felder des Arbeitsbandes außerhalb des durch die Begrenzungszeichen $\$$ und ¢ eingeschlossenen Abschnitts das Leerzeichen b enthalten. Wie immer bei Turingmaschinen, läßt sich δ als (partielle) Abbildung, die auf endlichen Mengen definiert ist, durch eine endliche Maschinentafel beschreiben.

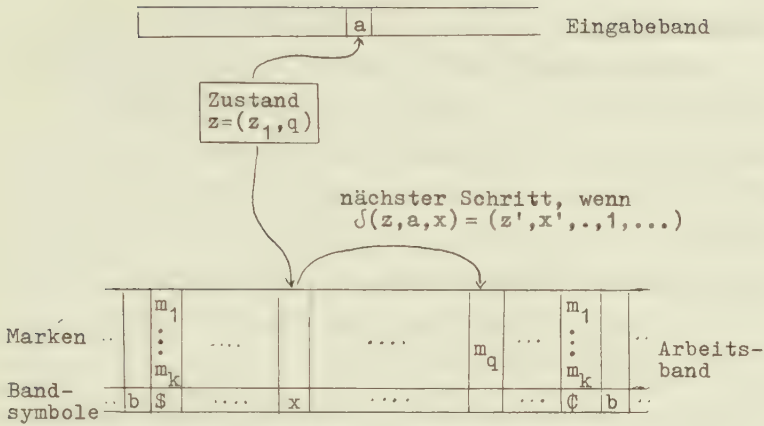


Abb. 2: Sprung einer TAS auf eine Marke m_q

Zur formalen Beschreibung des Verhaltens einer TAS kann in der üblichen Weise eine Konfiguration definiert werden als ein Quintupel $\alpha = (z, w, v, i, j)$ mit $z \in Z$, $w = a^{(1)} \dots a^{(n)}$, $w \in A^+$, $v = (M^{(1)}, x^{(1)}) \dots (M^{(s)}, x^{(s)}) \in (M, \$)(\mathcal{P}(M) \times X)^*(M, \$)$, $i, j \in \mathbb{N}$, $1 \leq i \leq n+1$, $1 \leq j \leq |v|$. Dadurch wird beschrieben, daß die TAS sich im Zustand z befindet, auf dem Eingabeband steht die Folge w , auf dem nichtleeren Teil des Arbeitsbandes im p -ten Feld ($1 \leq p \leq s$) jeweils die Markenmenge $M^{(p)}$ (die auch leer sein kann) und das Symbol $x^{(p)}$, die Leseköpfe befinden sich über dem i -ten Symbol von w bzw. über dem j -ten Marken-Symbol-Paar von v . Damit lassen sich die Bewegungen der TAS formal als Aufeinanderfolgen von Konfigurationen beschreiben, was durch $\alpha \vdash \beta$ wiedergegeben wird. Die Folgekonfiguration β ist dabei durch α und die Überföhrungsfunktion J eindeutig festgelegt. Hier soll das nur für die Sprungoperation ausgeföhrert werden, die bei der normalen TM nicht vorkommt:

Sprung nach links:

$$(z, w \leftarrow, v, i, j) \vdash (z', w \leftarrow, v', i', j')$$

mit $v = (M^{(1)}, x^{(1)}) \dots (M^{(j)}, x^{(j)}) \dots (M^{(s)}, x^{(s)})$

und $v' = (M^{(1)}, x^{(1)}) \dots (M^{(j-1)}, x^{(j-1)}) (M', x')$
 $(M^{(j+1)}, x^{(j+1)}) \dots (M^{(s)}, x^{(s)})$

sowie $i' = i + d_1$,

falls $\delta(z, a^{(i)}, x^{(j)}) = (z', x', d_1, -1, j_1, \dots, j_k)$

$z = (z_1, z_2)$ mit $z_2 = q > 0$, $m_q \in M^{(j')}$ ($j' < j$)

und $m_q \notin M^{(j-1)}, \dots, M^{(j'+1)}$.

Die dabei an die Stelle von $M^{(j)}$ tretende neue Markenmenge M' ist

$$M' = (M^{(j)} \cup \{m_p \mid j_p = 1\}) \setminus \{m_p \mid j_p = -1\}.$$

Analog ist das Aufeinanderfolgen von Konfigurationen für den Sprung nach rechts definiert.

Wie üblich werden die transitive Hülle der Relation " $\vdash$ ", Endkonfiguration und akzeptierte Wortmenge einer TAS definiert: Für zwei Konfigurationen α und β gilt $\alpha \vdash^* \beta$, wenn es Konfigurationen $\alpha_1, \dots, \alpha_t$ mit $\alpha = \alpha_1$, $\beta = \alpha_t$ und $\alpha_i \vdash \alpha_{i+1}$ ($i = 1, \dots, t-1$) gibt. Endkonfiguration einer TAS ist eine Konfiguration $\alpha = (z, w \leftarrow, v, i, j)$, wenn $\delta(z, a^{(i)}, x^{(j)})$ nicht definiert ist. Gilt für eine Kette $w \in A^+$ und eine TAS $\mathcal{T}$ $\alpha \vdash^* \beta$ mit $\alpha = (z_0, w \leftarrow, (M, \$)(M, \phi), 1, 1)$ (Konfiguration mit Startzustand z_0 , w als Eingabe und leerem Arbeitsband), $\beta = (z, w \leftarrow, v, |w| + 1, j)$, wobei β Endkonfiguration ist, so heißt w von $\mathcal{T}$ akzeptiert, falls $z \in Z_F$, und abgelehnt, falls $z \notin Z_F$. Maschinen, die die gleiche akzeptierte Wortmenge besitzen, heißen äquivalent.

Zeit- und Bandbeschränkung werden für die TAS analog zum Vorgehen bei einer TM definiert: Eine monotone Funktion $T: \mathbb{N} \rightarrow \mathbb{N}$ mit $T(n) \geq n$ heißt Zeitfunktion. Ist eine TAS $\mathcal{T}$ und eine Zeitfunktion $T(n)$ gegeben, so arbeitet $\mathcal{T}$ mit Zeitbeschränkung $T(n)$, wenn für alle Folgen von Konfigurationen $\alpha_1 \vdash \dots \vdash \alpha_t$ mit $\alpha_1 = (z_0, w\downarrow, (M, \$)(M, \emptyset), 1, 1)$, $|w| = n$ gilt: $t \leq T(n)$. Analog werden Bandfunktionen $L(n)$ definiert, eine TAS arbeitet mit Bandbeschränkung $L(n)$, wenn für alle Konfigurationen $\alpha_p = (z, w\downarrow, v, i, j)$, $|w| = n$, $v \in (M, \$)(\mathcal{P}(M) \times X)^{1-2}(M, \emptyset)$ in der Folge $\alpha_1 \vdash \dots \vdash \alpha_t$ (α_1 wie oben) gilt: $1 \leq L(n)$.

Zusätzlich zur TAS soll nun noch ein weiteres Maschinenmodell, das der Turingmaschine mit einseitig begrenztem assoziativem Speicherzugriff (1TAS), eingeführt werden. Die 1TAS läßt sich formal in der gleichen Weise wie die TAS als 7-Tupel $\mathcal{T} = (A, X, M, Z, Z_P, z_0, \delta)$ beschreiben, das Arbeitsband der 1TAS ist jedoch auf einer Seite (o.B.d.A. links) begrenzt, und der wesentliche Unterschied zur TAS liegt darin, daß ein Sprung der 1TAS unabhängig von der aktuellen Stellung des Lesekopfes immer zur ersten Marke des entsprechenden Typs (vom Bandanfang aus gezählt) führt. Dementsprechend ist das erste Feld des Bandes nur noch mit einem Begrenzungszeichen $\$$ (das nicht verschoben werden darf), nicht unbedingt aber mit einer Marke versehen. Für aufeinanderfolgende Konfigurationen gilt im Fall eines Sprunges:

$$(z, w\downarrow, v, i, j) \vdash (z', w\downarrow, v', i', j')$$

mit $v = (M^{(1)}, x^{(1)}) \dots (M^{(j)}, x^{(j)}) \dots (M^{(s)}, x^{(s)})$

und $v' = (M^{(1)}, x^{(1)}) \dots (M^{(j-1)}, x^{(j-1)}) (M', x')$
 $(M^{(j+1)}, x^{(j+1)}) \dots (M^{(s)}, x^{(s)})$,

falls $\delta(z, a^{(i)}, x^{(j)}) = (z', x', \dots, d_2, \dots)$, $d_2 \in \{-1, 0, 1\}$
 beliebig, $z = (z_1, z_2)$ mit $z_2 = q > 0$, $m_q \in M^{(j')}$
 und $m_q \in M^{(1)}, \dots, M^{(j'-1)}$ (hierin drückt sich der
 wesentliche Unterschied zur TAS aus)

(Bezeichnungen wie S. 11/12)

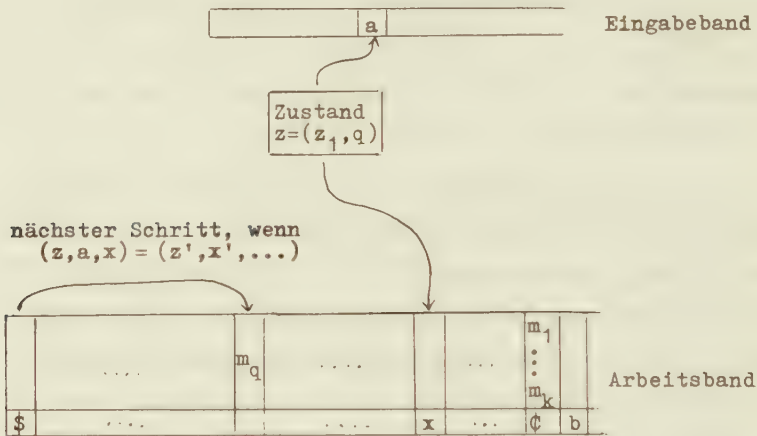


Abb.3: Sprung einer 1TAS auf eine Marke m_q

Das Problem, aus mehreren Marken gleichen Typs das Sprungziel für den nächsten Schritt auszuwählen, entspricht dem, was in der technisch orientierten Literatur über assoziative Speicherung das Problem der "multiple responses" genannt wird. TAS und 1TAS entsprechen in diesem Sinn zwei Lösungsmöglichkeiten für dieses Problem, wobei die TAS die allgemeinere Lösung bietet (Wie man sich leicht überzeugt, läßt sich jede 1TAS durch eine TAS mit einer zusätzlichen Marke in Linearzeit simulieren).

Obwohl es auf den ersten Blick so scheinen könnte, als sei es bei der 1TAS sinnlos, eine Marke öfter als einmal auf dem Band zu setzen, sind doch auch hier Sprungfolgen unter Verwendung mehrerer Marken desselben Typs möglich: Wenn die Maschine nach einem Sprung auf eine Marke m diese löscht und statt dessen eine neue Marke m' setzt, so erfolgt der nächste Sprung automatisch auf die zweite Marke m :

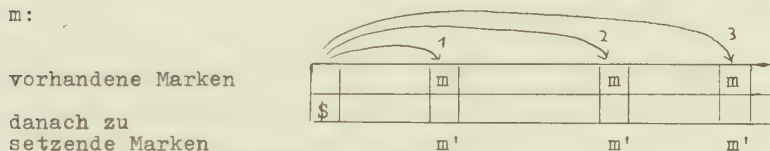


Abb. 4: Mögliche Sprungfolge einer 1TAS

Bei einem späteren zweiten Durchgang durch die entsprechenden Felder kann dann die Marke m' die Rolle von m spielen, sie wird nach einem Sprung wieder durch m ersetzt usw. Diese Technik des Springens wird in einigen folgenden Beweisen für die 1TAS benutzt.

Das Speed-Up-Theorem für die Bandkomplexität gilt für die TAS und 1TAS ebenso wie für die TM:

Lemma 2.1: Zu jeder TAS (1TAS) $\mathcal{T}$ mit Bandbeschränkung $L(n)$ und zu jeder Konstanten $C > 0$ gibt es eine äquivalente TAS (1TAS) $\mathcal{T}'$ mit Bandbeschränkung $C.L(n)$.

Beweis: Der Beweis kann ebenso geführt werden wie im Fall der TM (siehe z.B. [HU 69]): Sei r eine Konstante, so daß $r.C \geq 2$ gilt. Dann wird $\mathcal{T}'$ so konstruiert, daß ein Bandfeld von $\mathcal{T}'$ jeweils den Inhalt von r nebeneinanderliegenden Bandfeldern von $\mathcal{T}$ speichert, wobei zusätzlich noch in geeigneter Codierung die Information enthalten sein soll,

welches der r Bandfelder von $\mathcal{T}$ mit welchen Marken markiert war. Ein Bandfeld von $\mathcal{T}'$ wird mit der Vereinigung der Marken der entsprechenden Bandfelder von $\mathcal{T}$ markiert. Die Simulation auch eines Sprunges von $\mathcal{T}$ ist dann kein Problem, da $\mathcal{T}'$ im endlichen Gedächtnis zunächst feststellen kann, ob der Sprung vom $\mathcal{T}$ bei $\mathcal{T}'$ etwa innerhalb des von $\mathcal{T}'$ betrachteten ("größeren") Bandfeldes bleibt oder ob auch von $\mathcal{T}'$ ein Sprung auszuführen ist.

Dagegen ist die Übertragung der Beweisidee für das Speed-Up-Theorem bei der Zeitkomplexität nicht möglich, da zwei aufeinanderfolgende Schritte der TAS beliebig weit auseinanderliegende Felder des Bandes berühren können, also nicht durch Zusammenfassen mehrerer Felder zu einem Schritt zusammengefaßt werden können.

Im Hinblick vor allem auf den folgenden Abschnitt 3, der sich mit On-Line-Spracherkennung durch die TAS beschäftigt, wurde hier das Modell einer TAS mit eigenem Eingabeband beschrieben. Analog läßt sich auch eine TAS ohne eigenes Eingabeband oder eine mehrbändige TAS formal definieren. Für eine r -bändige TAS wäre dabei die Markenkomponente des Zustands wiederum als r -Tupel $(z^{(1)}, \dots, z^{(r)})$ aufzufassen, wobei ein Sprung auf dem i -ten Band dann jeweils von $z^{(i)}$ abhängt. Das folgende Lemma zeigt jedoch, daß im Fall der TAS ein Übergang zu einem Modell mit mehreren Bändern keine wesentliche Zeitersparnis bringt.

Lemma 2.2: Zu jeder r -bändigen TAS (1TAS) $\mathcal{T}$ mit k Marken und Zeitbeschränkung $T(n)$ gibt es eine äquivalente ein-

bändige TAS (1TAS) $\mathcal{T}'$ mit $k.r+2$ Marken, die mit Zeitbeschränkung $(4r+1).T(n)$ arbeitet.

Beweis: $\mathcal{T}'$ simuliert $\mathcal{T}$, indem der Inhalt der Bänder von $\mathcal{T}$ bei $\mathcal{T}'$ auf einem Band mit entsprechend vergrößertem Bandalphabet und vergrößerter Markenmenge gespeichert wird. Hat $\mathcal{T}$ r Bänder und steht im j -ten Feld des p -ten Bandes ($1 \leq p \leq r$) jeweils das Symbol x_{i_p} und die Marken $m_{i_p,1}, \dots, m_{i_p,n_p}$, so stehen im j -ten Feld von $\mathcal{T}'$ das Symbol $(x_{i_1}, \dots, x_{i_r})$ und die Marken $m_{i_1,1}^1, \dots, m_{i_1,n_1}^1, m_{i_2,1}^2, \dots, m_{i_2,n_2}^2, \dots, m_{i_r,1}^r, \dots, m_{i_r,n_r}^r$. Denn um alle Sprünge von $\mathcal{T}$ richtig simulieren zu können, muß $\mathcal{T}'$ für gleiche Marken, die auf verschiedenen Bändern von $\mathcal{T}$ vorkommen, verschiedene Marken verwenden. Außerdem wird das Feld, auf dem bei $\mathcal{T}$ jeweils der i -te Lesekopf steht, bei $\mathcal{T}'$ in einer weiteren Symbolspur mit einem zusätzlichen Symbol s_i gekennzeichnet; zudem werden diese Felder noch mit einer zusätzlichen Marke m_L (L =Lesekopf) versehen. Zur Simulation eines Schrittes von

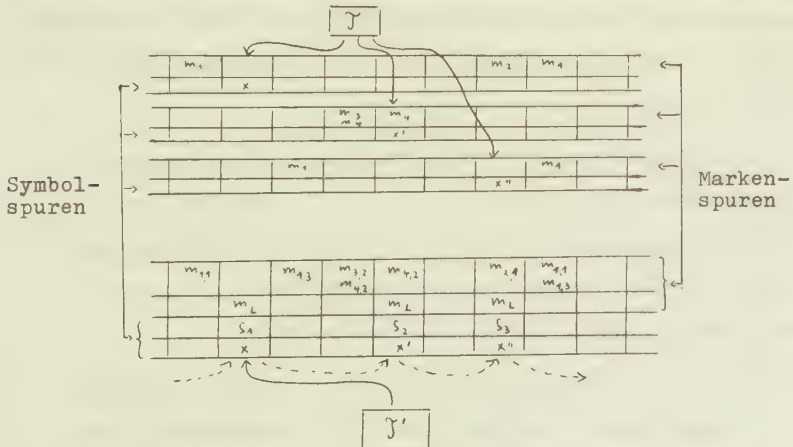


Abb.5: Simulation einer 3-Band-TAS durch eine 1-Band-TAS

$\mathcal{J}$ springt dann der Lesekopf von $\mathcal{J}'$, von einem Bandbegrenzungszeichen ausgehend, zunächst nacheinander auf die Felder mit der Marke m_L , liest, welches Symbol unter welchem Kopf von $\mathcal{J}$ stünde und ersetzt die Marke m_L durch eine weitere Marke m_L , ($r+1$ Schritte). In einem weiteren Durchgang springt er wieder auf die einzelnen, nun mit m_L , markierten Felder, beschreibt sie entsprechend dem Schritt von $\mathcal{J}$ neu, löscht m_L , und setzt das jeweilige Symbol s_i und die Marke m_L gemäß dem Schritt (evtl. auch Sprung) des Kopfes auf dem i -ten Band von $\mathcal{J}$. Wegen der dabei evtl. zu simulierenden Sprünge von $\mathcal{J}$ kann es für die TAS $\mathcal{J}'$ nötig sein, jeweils vom Bandende aus die nächste noch nicht erfaßte Marke m_L zu suchen; bei der 1TAS erfolgen Sprünge ohnehin immer vom Bandanfang aus. $\mathcal{J}'$ kommt für diese Operationen mit höchstens $3r$ Schritten aus, danach steht der Lesekopf wieder an einem Bandende zur Simulation des nächsten Schrittes von $\mathcal{J}$ bereit. Die Simulation eines Schrittes von $\mathcal{J}$ erfordert damit höchstens $4r+1$ Schritte von $\mathcal{J}'$, $\mathcal{J}'$ arbeitet daher mit Zeitbeschränkung $(4r+1) \cdot T(n)$; außerdem kommt $\mathcal{J}'$ zur Simulation mit $k \cdot r+2$ Marken aus.

Eine entsprechende Aussage läßt sich auch für das Modell einer k -köpfigen TAS bzw. 1TAS mit mehreren Köpfen auf einem Band machen, auch hier bringt der Übergang zu einer TAS bzw. 1TAS mit nur einem Kopf auf einem Band keine wesentliche Erhöhung der Komplexität.

Da jede mehrbändige TM auch als TAS (mit $M = \emptyset$) aufgefaßt werden kann, folgt aus Lemma 2.2 insbesondere, daß es zu jeder mehrbändigen TM eine äquivalente einbändige

TAS mit einer bis auf einen konstanten Faktor gleichen Zeitbeschränkung gibt. Die Frage liegt nahe, ob auch das Umgekehrte gilt, ob es also zu jeder einbändigen TAS mit Zeitbeschränkung $T(n)$ eine äquivalente mehrbändige TM mit Zeitbeschränkung $C.T(n)$ gibt. Für ein Modell der TAS mit stärker eingeschränktem Sprungvermögen haben das M.J.Fischer und A.L.Rosenberg gezeigt [FR 68]:

Satz 2.3: Sei $\mathcal{T}$ eine (ein- oder mehrbändige) TAS mit Zeitbeschränkung $T(n)$ und mit den folgenden Einschränkungen:

- (1) Jede Marke kann auf einem Band höchstens einmal gesetzt werden,
- (2) Eine einmal gesetzte Marke kann nicht wieder entfernt werden.

Dann gibt es zu $\mathcal{T}$ eine äquivalente mehrbändige TM $\mathcal{T}'$, die mit derselben Zeitbeschränkung $T(n)$ arbeitet.

Die Ergebnisse von Abschnitt 3 lassen erkennen, daß die wesentliche Einschränkung gegenüber der TAS vor allem in Bedingung (1) liegt.

3. Relative Leistungsfähigkeit von TAS, 1TAS und TM

(On-Line-Modell)

Will man durch einen konstruktiven Beweis, durch Definition einer geeigneten Sprache, die Leistungsfähigkeit zweier verschiedener Maschinenmodelle miteinander vergleichen, so stößt man auf die Schwierigkeit, daß kaum Beweismethoden bekannt sind, um untere Schranken für die Komplexität von Algorithmen (jedenfalls soweit es Spracherkennungsalgorithmen sind) nachzuweisen. Deshalb beschränkt sich dieser Abschnitt auf On-Line-Maschinen im Sinne von F.C.Hennie [He 66]. Es zeigt sich, daß unter dieser Einschränkung recht genaue Aussagen über die relative Leistungsfähigkeit von TM, 1TAS und TAS möglich sind.

Zunächst soll definiert werden, was unter dem On-Line-Modell einer TM (1TAS, TAS) verstanden werden soll:

Eine On-Line-TM (-1TAS, -TAS) ist eine TM (1TAS, TAS) mit einem Eingabeband, von dem nur gelesen werden kann und auf dem sich der Lesekopf nur in einer Richtung bewegen kann, und einem oder mehreren Arbeitsbändern, für die gilt: Das i -te Symbol a_i auf dem Eingabeband darf erst eingelesen werden, wenn die Maschine für das Wort $a_1 \dots a_{i-1}$ entschieden hat (etwa - bei geeignet modifizierter Definition des Maschinenmodells - durch eine entsprechende Ausgabe), ob es zur akzeptierten Wortmenge der Maschine gehört oder nicht.

Dies entspricht der in der Literatur üblichen Definition ([He 66],[CA 69]).

Allgemeiner soll von einer On-Line-Maschine gesprochen werden, wenn eine Aussage in gleicher Weise für das Modell der On-Line-TM, -1TAS oder -TAS gilt.

Von Hennie wurde in [He 66] erstmals eine nichttriviale untere Schranke für die Erkennung einer gewissen Sprache durch eine On-Line-TM angegeben. Dabei beruht der Beweis wesentlich auf der Tatsache, daß eine TM mit r Arbeitsbändern in t Schritten nur Zugriff zu $(2t+1) \cdot r$ Feldern auf den Bändern hat. Eine ähnliche Aussage läßt sich auch für die 1TAS machen, dazu soll zunächst allgemein definiert werden:

Als Bandkonfiguration β einer Maschine wird der jeweils aktuelle Inhalt der Arbeitsbänder einschließlich der Kopfstellungen und eventueller Marken (bei TAS oder 1TAS) bezeichnet; für eine TAS ist also, wenn $\alpha = (z, w, v, i, j)$ eine Konfiguration im Sinne der Definition von S. 11 bezeichnet, das Paar $\beta = (v, j)$ (dritte und fünfte Komponente von α) die Bandkonfiguration. Zu einer gegebenen Bandkonfiguration β und einer Maschine $\mathcal{J}$ heißt ein Bandfeld b in t Schritten erreichbar, wenn es eine Folge von nicht mehr als t Schritten von $\mathcal{J}$ gibt, die $\mathcal{J}$, von β ausgehend, in eine Konfiguration überführt, bei der ein Lesekopf von $\mathcal{J}$ über dem Feld b steht. Als Klasse von Maschinen wird die Menge aller Maschinen, die durch eine gemeinsame Definition gegeben sind, bezeichnet, etwa die Klasse der r -Band-Turingmaschine oder die Klasse aller TAS's mit k Marken. Eine Klasse $\mathcal{K}$ von Maschinen hat Speicherzugriff von der Ordnung k , wenn es eine Konstante C gibt, so daß für jede Bandkonfiguration β

die für eine Maschine aus $\mathcal{K}$ möglich ist, und für alle Maschinen aus $\mathcal{K}$ gilt: Für die Vereinigung F aller Bandfelder, die von Maschinen der Klasse $\mathcal{K}$, von der Bandkonfiguration β ausgehend, in t Schritten erreichbar sind, gilt $|F| \leq C.t^k$.

Lemma 3.1: Die Klasse der TAS's mit einer Marke sowie die Klasse der 1TAS mit fester Markenzahl - hier unabhängig von der Anzahl der Marken - haben Speicherzugriff von der Ordnung 2.

Beweis: Es wird für die 1TAS bewiesen, daß für jede gegebene Bandkonfiguration die Anzahl der durch verschiedene Maschinen mit k Marken in t Schritten erreichbaren Felder durch $k.t^2 + 2.t + 1$ beschränkt ist. Für die TAS mit einer Marke kann dabei der erste Teil des Beweises übernommen werden, die Anzahl der erreichbaren Felder ist hier $2.t^2 + 2.t + 1$.

a) Einzelschritte und Sprünge auf nur eine Marke: Führt eine 1TAS $\mathcal{T}$ keine Sprünge aus, so können in t Schritten höchstens $2t+1$ Felder (das Ausgangsfeld eingeschlossen) erreicht werden. Ist der erste Schritt ein Sprung auf eine Marke m , so sind vom Sprungziel aus weitere $2(t-1)+1$ Felder durch Schritte ohne Sprung zu erreichen. Wird nach dem Sprung m gelöscht, so daß ein zweiter Sprung auf die nächste Marke m führt, so werden bis zu $2(t-2)+1$ weitere Felder erreichbar usw., insgesamt sind das bis zu $2t+1$ Felder, die durch Einzelschritte, und $2.(t-1+t-2+\dots+1)+t = t^2$ weitere Felder, die durch mindestens einen Sprung und anschließende weitere Sprung oder Einzelschritte erreichbar sind.

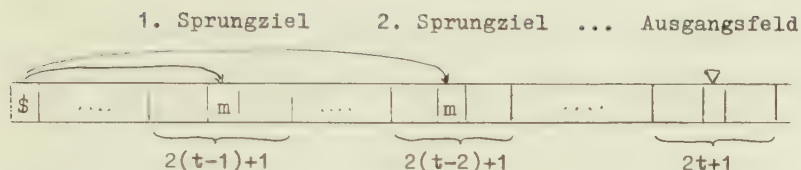


Abb.6: Erreichbare Felder für die 1TAS (eine Marke)

Für jede weitere Marke m' gibt es in derselben Weise bis zu t^2 verschiedene Felder, die durch Sprünge auf m' und anschließende Schritte erreichbar sind.

Ist $\mathcal{T}$ eine TAS mit einer Marke, so gelten die gleichen Überlegungen wie oben, nur mit dem Unterschied, daß hier die zunächst erreichbaren Marken m nicht gelöscht werden müssen, wenn anschließend auf weiter entfernt liegende Marken gesprungen werden soll, und daß vom Ausgangsfeld aus Sprünge in beiden Richtungen möglich sind. Die Anzahl der erreichbaren Bandfelder beträgt hier also $2 \cdot t^2 + 2 \cdot t + 1$. Der Beweis für die TAS mit einer Marke ist damit schon geführt, für die 1TAS sind noch weitere mögliche Fälle zu berücksichtigen.

b) Aufeinanderfolgende Sprünge auf verschiedene Marken:

Nun soll gezeigt werden, daß durch Sprünge einer 1TAS gar keine anderen als die in a) schon gezählten Felder ($k \cdot t^2$ für k Marken und t Schritte) erreichbar sind, auch wenn Sprünge auf verschiedene Marken, Setzen und Löschen von Marken beliebig zugelassen sind. Sei dazu F_1^i die Menge aller Felder, die von $\mathcal{T}$ in i Schritten erreichbar sind, wobei etwaige Sprünge innerhalb dieser i Schritte nur auf Marken desselben Typs erfolgen, und sei F_1 die Menge der in i Schritten überhaupt erreichbaren Felder. Nach der obigen

Überlegung gilt dann $|F'_i| = k \cdot i^2 + 2 \cdot i + 1$, außerdem gilt selbstverständlich $F_i \subseteq F_{i+1}$, $F'_i \subseteq F'_{i+1}$ und $F'_i \subseteq F_i$ für alle i . Nun soll gezeigt werden, daß γ in i Schritten überhaupt keine anderen Felder als die Felder von F'_i erreichen kann, daß also $F_i = F'_i$ gilt:

b1) Wird nach Sprüngen auf Marken m oder nach Einzelschritten im i -ten Schritt ein Sprung auf eine Marke $m' \neq m$ ausgeführt, ohne daß Marken neu gesetzt oder gelöscht wurden, so wird dadurch F_i gegenüber F'_i nicht vergrößert, da das Feld mit der Marke m' schon in F'_i enthalten ist.

Damit gilt unter der Voraussetzung, daß keine Marken neu gesetzt oder gelöscht werden, $F_i = F'_i$ für alle i . Andererseits kann gezeigt werden, daß auch das Setzen oder Löschen von Marken im i -ten Schritt einer TAS für keinen späteren j -ten Schritt ($j > i$) die Menge F_j gegenüber F'_j vergrößern kann:

b2) Wird im i -ten Schritt eine Marke zusätzlich gesetzt, so wird im $(i+1)$ -ten, $(i+2)$ -ten usw. Schritt dadurch ein Feld eventuell durch Sprung erreichbar, auf dem der Lesekopf aber schon nach dem i -ten Schritt stehen kann, F_{i+1} , F_{i+2} , ... wird also gegenüber F_i dadurch nicht vergrößert.

b3) Wird im i -ten Schritt (der etwa ein Sprung auf eine Marke m' oder ein Einzelschritt sein) auf einem Feld f eine Marke m gelöscht, die die erste Marke ihres Typs auf dem Band ist, so wird dadurch im $(i+1)$ -ten Schritt ein weiteres mit m markiertes Feld f' erreichbar. Dieses Feld wäre aber auch durch zwei aufeinanderfolgende Sprünge auf die Marke m (mit Löschen nach dem ersten Sprung) erreichbar

gewesen, f' ist also in F'_{i+1} enthalten, wenn nur f - wie vorausgesetzt - schon in F'_i enthalten war.

b4) Es werde im i -ten Schritt auf dem Feld f eine Marke m gelöscht, die die p -te Marke ihres Typs auf dem Band ist. Dadurch wird das Feld f' mit der $(p+1)$ -ten Marke m in p statt $p+1$ weiteren Schritten (in der Weise, wie in a) geschildert) erreichbar, insgesamt wird damit f' ein Feld aus der Menge F_{i+p} . Andererseits wäre f' auch durch $p+1$ Sprünge auf Marken m im ersten, zweiten, ..., $(p+1)$ -ten Schritt erreichbar, f' ist daher in F'_{p+1} enthalten; wegen $i \geq 1$ gilt aber $F'_{p+1} \subseteq F'_{i+p}$. Die Menge F_{i+p} wird also durch diese Möglichkeit des Löschens von Marken gegenüber F'_{i+p} nicht vergrößert.

Da b1) - b4) alle gegenüber a) zusätzlichen Möglichkeiten ausschöpfen, folgt insgesamt die Behauptung des Lemmas.

Definiert man in der üblichen Weise eine Turingmaschine mit mehrdimensionalem Speicher, wie dies etwa in [HS 65] geschieht, so gilt offensichtlich (vgl. [CA 69]):

Lemma 3.2: Die Klasse der Turingmaschinen mit k -dimensionalem Speicherband hat Speicherzugriff von der Ordnung k .

Dies gilt offenbar auch dann, wenn zugelassen wird, daß mehrere Leseköpfe der TM im mehrdimensionalen Speicher gleichzeitig operieren (so das Modell von [St 71]).

Speicherzugriff von der Ordnung k und Zeitbeschränkung für On-Line-Maschinen lassen sich durch den folgenden Satz miteinander verbinden:

Satz 3.3: Für jedes $k \in \mathbb{N}$ ($k \geq 1$) gibt es eine Sprache L_k ,

so daß gilt:

- (1) Wird L_k von einer On-Line-Maschine $\mathcal{T}$ mit Speicherzugriff von der Ordnung k akzeptiert, die mit Zeitbeschränkung $T(n)$ arbeitet, so gilt

$$T(n) \geq C \cdot n^{1 + \frac{1}{k \cdot (k+1)}}$$

für unendlich viele n und für ein $C > 0$.

- (2) L_k kann von einer On-Line-TAS $\mathcal{T}_1$ mit einem Arbeitsband und k Marken in Linearzeit akzeptiert werden.

Verfügt $\mathcal{T}_1$ dazu noch über ein zusätzliches Zählerband oder hat $\mathcal{T}_1$ zwei weitere Marken, insgesamt also $k+2$ Marken, so kann für $\mathcal{T}_1$ die zusätzliche Einschränkung gelten, daß Marken auf dem Arbeitsband nur gesetzt, nicht aber gelöscht werden dürfen.

- (3) Für $k=1$ gilt: L_1 kann von einer On-Line-1TAS $\mathcal{T}_2$ mit einem Arbeitsband und zwei Marken in Linearzeit akzeptiert werden.

- (4) L_k kann von einer On-Line-TM $\mathcal{T}_3$ mit $(k+1)$ -dimensionalem Speicher in Linearzeit akzeptiert werden.

Beweis: Die Beweisidee von [He 66] wird geeignet modifiziert: L_k wird definiert als eine Menge von Wörtern, die jeweils aus einem "Anfangsabschnitt" u und einem "Endabschnitt" v bestehen. Dabei ist u eine Folge von m^k Blöcken aus $\{0,1\}^m$ der Länge m (m variabel, aber für jedes $w \in L_k$ fest), v besteht aus weiteren Blöcken mit Länge $\leq (k+1) \cdot m + k$, wobei die m letzten Zeichen mit einem bestimmten Block im Abschnitt u übereinstimmen. Formal wird L_k in der folgenden Weise beschrieben:

Für $w \in \{0, 1, c_1, c_2, \dots, c_{k+1}\}^*$ gilt $w \in L_k$ genau dann, wenn gilt:

(i) $w = u c_{k+1} v$ (u "Anfangsabschnitt", v "Endabschnitt"),
wobei

$$\begin{aligned}
 u = & \underbrace{u_1, \dots, 1, 1}_{k \text{ Indizes}} c_1 u_1, \dots, 1, 2 c_1 u_1, \dots, 1, 3 c_1 \dots u_1, \dots, 1, m c_2 \\
 & u_1, \dots, 1, 2, 1 c_1 u_1, \dots, 1, 2, 2 c_1 \dots u_1, \dots, 1, 2, m c_2 \\
 & \dots \\
 & u_1, \dots, 1, m, 1 c_1 u_1, \dots, 1, m, 2 c_1 \dots u_1, \dots, 1, m, m c_3 \\
 & u_1, \dots, 1, 2, 1, 1 c_1 u_1, \dots, 1, 2, 1, 2 c_1 \dots u_1, \dots, 1, 2, 1, m c_2 \\
 & \dots \\
 & \dots \\
 & \dots \\
 & u_m, \dots, m, 1 c_1 u_m, \dots, m, 2 c_1 u_m, \dots, m, 3 c_1 \dots u_m, \dots, m, m,
 \end{aligned}$$

$$v = v_1 c_k v_2 c_k \dots c_k v_t c_k$$

mit

$$\begin{aligned}
 & u_{i_1, \dots, i_k} \in \{0, 1\}^* \\
 & |u_{i_1, \dots, i_k}| = m \quad \text{für alle } (i_1, \dots, i_k) \in \{1, \dots, m\}^k \\
 & |v_p| \leq (k+1) \cdot m + k \\
 & v_p = 1^{i_p, 1} c_1 1^{i_p, 2} c_1 \dots c_1 1^{i_p, k} c_1 x_p \quad \left. \vphantom{v_p} \right\} \text{ für alle } p = 1, \dots, t \\
 & \text{mit } 1 \leq i_p, j \leq m \quad (j = 1, \dots, k), \quad x_p \in \{0, 1\}^*, \quad |x_p| = m \\
 & (v_p \text{ enthält also unäre Darstellungen von } k \text{ Indizes } \leq m \\
 & \text{und anschließend die Kette } x_p)
 \end{aligned}$$

(ii) der letzte Block v_t von v erfüllt die Bedingung

$$x_t = u_{i_t, 1, i_t, 2, \dots, i_t, k},$$

er ist also gleich dem durch $i_t, 1, \dots, i_t, k$ bezeichneten Block im Anfangsabschnitt u .

Wenn $t = C \cdot m^k$ mit einer Konstanten C gilt, dann sind also n (die Gesamtlänge des Wortes w) und m durch die Beziehung $n = C' \cdot m^{k+1}$ bzw. $m = C'' \cdot \sqrt[k+1]{n}$ (C', C'' geeignete Konstanten) miteinander verknüpft.

In der bei der Beschreibung von Datenstrukturen üblichen Ausdrucksweise könnte man sagen: Ein Anfangsabschnitt u realisiert eine Speicherung von m^k Datenelementen (hier Blöcke genannt) mit Hilfe einer k -stufigen Indexhierarchie, das Erkennen eines Satzes aus L_k entspricht bei gegebenem Endabschnitt v einer indexsequentiellen Suche im Anfangsabschnitt u .

Der Beweis von (1) wird nun in mehreren Schritten geführt:

a) Zunächst kann man leicht einsehen, daß L_k von einer On-Line-Mehrband-TM mit Zeitbeschränkung $C_1 \cdot n^2$ akzeptiert werden kann: Die Maschine nimmt den Anfangsabschnitt u auf ein Arbeitsband, sucht beim Einlesen eines neuen Blockes v_p des Endabschnitts nach dem durch die Indizes $i_{p,1}, \dots, i_{p,k}$ bezeichneten Block im Anfangsabschnitt und vergleicht ihn mit x_p ; sie benötigt daher pro Block des Endabschnitts höchstens m^{k+1} Schritte. Die übrigen Bedingungen für die Kette w (gleiche Länge für die einzelnen Blöcke usw.) sind in Linearzeit überprüfbar, als grobe Abschätzung gilt daher für die Maschine die Zeitbeschränkung $C_1 \cdot n^2$.

b) Sei nun $A_{k,m}$ für beliebiges, aber festes m die Menge aller Worte w , die (i) erfüllen und für die $t = m^k$ gilt, damit also für die Gesamtlänge $n = n_{k,m} = O(m^{k+1})$. Im folgenden soll gezeigt werden, daß es einen Anfangsabschnitt u und einen ersten Block v_1 des Endabschnitts eines Wortes

$w \in A_{k,m}$ gibt, so daß eine On-Line-Maschine $\mathcal{T}$ mit Speicherzugriff von der Ordnung k , die L_k akzeptiert, bei der Bearbeitung von v_1 mindestens $C.m.\sqrt[k]{m}$ Schritte braucht. Ist eine Konfiguration von $\mathcal{T}$ in der üblichen Weise (wie für die TAS in Abschnitt 1 angegeben) definiert, so gilt: Das Einlesen zweier verschiedener Anfangsabschnitte u, u' von Worten $w, w' \in A_{k,m}$ muß bei $\mathcal{T}$ zu zwei verschiedenen Konfigurationen führen. Denn sei in u, u' etwa $u_{i_1}, \dots, i_k \neq u'_{i_1}, \dots, i_k$. Dann läßt sich ein Block v_1 des Endabschnitts angeben (nämlich $v_1 = 1^{i_1} c_1 1^{i_2} c_1 \dots c_1 1^{i_k} c_1 u_{i_1}, \dots, i_k$), für den $\mathcal{T}$ verschieden reagieren muß: Annahme für $u c_{k+1} v_1$, Ablehnung für $u' c_{k+1} v_1$. Es muß also mindestens $2^{m^{k+1}}$ verschiedene Konfigurationen geben, die in dieser Weise für die verschiedenen möglichen Anfangsabschnitte u charakteristisch sind. Um diese Beziehung zu präzisieren, wird für eine Maschine $\mathcal{T}$, die L_k akzeptiert, und für eine Wortmenge $A_{k,m}$ die Menge F aller Abbildungen

$$f: \{ \text{Anfangsabschnitte } u \text{ von } w \in A_{k,m} \} \\ \longrightarrow \{ \text{Konfigurationen einer Maschine } \mathcal{T} \}$$

betrachtet, für die gilt:

- 1) Es gibt ein $C_2 > 0$, so daß bei jeder Konfiguration, die als Bild unter einer Abbildung $f \in F$ auftritt, höchstens $C_2.n^2$ Felder auf jedem Band beschrieben sind,
- 2) Für jeden Block v_p aus $\{ 1^{i_{p,1}} c_1 1^{i_{p,2}} c_1 \dots c_1 1^{i_{p,k}} c_1 x_p \mid 1 \leq i_{p,j} \leq m, |x_p| = m, x_p \in \{0,1\}^* \}$ arbeitet $\mathcal{T}$ korrekt, d.h. von der Konfiguration $f(u)$ ausgehend entscheidet $\mathcal{T}$ richtig, wenn $v_p = 1^{i_{p,1}} c_1 1^{i_{p,2}} c_1 \dots c_1 1^{i_{p,k}} c_1 x_p$ als Block im Endabschnitt gegeben ist, ob $u_{i_{p,1}}, \dots, i_{p,k} = x_p$ gilt.

Jede Maschine $\mathcal{T}$, die L_k akzeptiert, realisiert also - vorausgesetzt, 2) ist erfüllt - eine solche Abbildung f ; im folgenden wird allerdings nicht die Zeit betrachtet, die bei gegebenem Anfangsabschnitt u zur Herstellung der Konfiguration $f(u)$ nötig ist, sondern die Zeit, die $\mathcal{T}$, von der Konfiguration $f(u)$ ausgehend, anschließend noch braucht. Sei dazu einer Abbildung $f \in F$ die "Prüfzeit" t_f zugeordnet als die maximale Anzahl von Schritten, die $\mathcal{T}$ machen muß, um zu beliebigem Anfangsabschnitt u und beliebigem Block v_p (v_p wie oben), von der Konfiguration $f(u)$ ausgehend zu entscheiden, ob $uc_{k+1} v_p \in L_k$ gilt. Sei t^* das Minimum aller Prüfzeiten für alle Abbildungen $f \in F$; die zugehörige Abbildung, bei der dieses Minimum erreicht wird, sei f^* . Eine solche "optimale" Abbildung mit minimaler Prüfzeit existiert sicher, da wegen 1) das Minimum nur über endlich viele Abbildungen gebildet wird und da wegen a) eine Abbildung, die 1) und 2) erfüllt, auch existiert.

Nach Voraussetzung ist $\mathcal{T}$ eine Maschine mit Speicherzugriff von der Ordnung k ; es gibt also eine Konstante C_3 , so daß von jeder Konfiguration von $\mathcal{T}$ aus in t Schritten höchstens $C_3 \cdot t^k$ Felder erreichbar sind. Die Anzahl der Konfigurationen, die sich für eine gegebene Stellung der Leseköpfe auf den Arbeitsbändern entweder im Zustand oder in der Inschrift auf einem dieser (pro Band) $C_3 \cdot t^k$ Felder voneinander unterscheiden, ist

$$q \cdot s^r \cdot C_3 \cdot t^k,$$

wenn q die Anzahl der Zustände, r die der Bänder und s die der Bandsymbole (einschließlich eventueller Markenkombina-

tionen im Fall der TAS oder 1TAS) von $\mathcal{J}$ ist. Setzt man nun

$$t = \left\lceil k \sqrt{\frac{m^{k+1} - \log q - 1}{C_3 \cdot r \cdot \log s}} \right\rceil,$$

so ergibt sich ¹⁾

$$\begin{aligned} r \cdot C_3 \cdot t^k &\leq q \cdot s \frac{m^{k+1} - \log q - 1}{\log s} \\ &= q \cdot 2^{m^{k+1} - \log q - 1} \\ &= 2^{m^{k+1} - 1} < 2^{m^{k+1}} - 1. \end{aligned}$$

Wie oben gezeigt, müssen aber verschiedene Anfangsabschnitte u, u' (von denen es insgesamt $2^{m^{k+1}}$ gibt) durch die Abbildung f^* auf verschiedene Konfigurationen abgebildet werden, und andererseits müssen nach Definition von t^* diese Konfigurationen sich im Zustand oder in Bandsymbolen unterscheiden, die in t^* Schritten erreichbar sind. Wäre $t^* < \left\lceil k \sqrt{\frac{m^{k+1} - \log q - 1}{\log s}} \right\rceil$, so gäbe es zwei verschiedene Anfangsabschnitte, die auf dieselbe Konfiguration abgebildet würden, im Widerspruch zu Bedingung 2). Insgesamt folgt daraus

$$t^* \geq \left\lceil k \sqrt{\frac{m^{k+1} - \log q - 1}{\log s}} \right\rceil.$$

c) Mit b) wurde also gezeigt, daß es für jede Maschine mit Speicherzugriff von der Ordnung k eine Konstante C_4 und eine Eingabefolge $w \in A_{k,m}$ gibt, so daß $\mathcal{J}$ zur Entscheidung, ob der erste Block des Endabschnittes von w zu einem Satz von L_k führt, mindestens $C_4 \cdot k \sqrt{m^{k+1}} = C_4 \cdot m \cdot k \sqrt{m}$ Schritte braucht. Nun ist noch zu zeigen, daß sich diese Abschätzung auch auf die anderen Blöcke des Endabschnitts übertragen

1) alle Logarithmen zur Basis 2, $\lfloor x \rfloor :=$ größte ganze Zahl $\leq x$.

läßt, daß also $\mathcal{T}$ nicht die gespeicherte Information so umordnen kann, daß die Entscheidung für die weiteren Blöcke schneller getroffen werden kann. Dazu kann, ebenso wie in [He 66], bewiesen werden, daß es eine Kette $w \in A_{k,m}$ gibt, so daß $\mathcal{T}$ bei der Eingabe von w mindestens $m^k \cdot t^*$ Schritte für die Bearbeitung des Endabschnitts braucht. Der Grundgedanke ist der, daß andernfalls bei geeigneter Wahl einer Abbildung $f \in F$ schon die Zeit für die Bearbeitung des ersten Blocks im Endabschnitt kleiner als t^* sein müßte.

Eine Konfiguration, in der sich $\mathcal{T}$ nach dem Einlesen eines Anfangsabschnitts u und einiger Blöcke $v_1, \dots, v_p$ des Endabschnitts einer Kette aus $A_{k,m}$ befindet, soll im folgenden als "schnelle Konfiguration" bezeichnet werden, wenn für alle denkbaren folgenden Blöcke die Prüfzeit kleiner als t^* ist, andernfalls als "langsame Konfiguration". Schon bewiesen wurde, daß für alle Anfangsabschnitte u von Worten $w \in A_{k,m}$ nach dem Einlesen nur des Anfangsabschnitts u die Maschine $\mathcal{T}$ sich auf alle Fälle in einer langsamen Konfiguration befinden muß. Angenommen, es gäbe für jeden Anfangsabschnitt u eine Folge von Blöcken im Endabschnitt, die jeweils zu einer schnellen Konfiguration α von $\mathcal{T}$ führt, so wäre dadurch eine Abbildung f' gegeben, die jeweils einem Anfangsabschnitt u eine Konfiguration $\alpha = f'(u)$ (die nach Annahme existierende Konfiguration) zuordnet. f' wäre damit eine der Abbildungen $f \in F$, unter denen f^* als optimal ausgewählt war. Die Prüfzeit t' für f' wäre dann aber, da $f'(u)$ für alle u eine schnelle Konfiguration wäre, geringer

als die für f^* , im Widerspruch zur Voraussetzung, daß f^* optimal sein sollte. Es muß also einen Anfangsabschnitt u und eine Folge von Blöcken $v_1, \dots, v_{m^k}$ geben, zu deren Prüfung $\mathcal{T}$ mindestens t^* Schritte bei jedem Block benötigt. Für dieses Wort benötigt $\mathcal{T}$ damit insgesamt $m^k \cdot t^* \leq C_4 \cdot m^k \cdot k \cdot \overline{m^{k+1}} = C_4 \cdot m^{k+1} \cdot k \sqrt{m} = C_5 \cdot n \cdot \frac{k(k+1)}{\sqrt{n}}$ Schritte für eine geeignete Konstante C_5 . Insgesamt ist damit Aussage (1) des Satzes bewiesen, daß eine Maschine $\mathcal{T}$ mit Speicherzugriff von der Ordnung k , die L_k akzeptiert, mit einer Zeitbeschränkung $T(n)$ arbeiten muß, für die

$$T(n) \geq C \cdot n \cdot \frac{k(k+1)}{\sqrt{n}} = C \cdot n^{1 + \frac{1}{k(k+1)}}$$

mit einer Konstanten $C > 0$ gilt.

Zu (2) läßt sich zunächst leicht angeben, wie eine TAS $\mathcal{T}_1$ mit k Marken die Blöcke des Endabschnitts eines Wortes aus L_k in Linearzeit überprüfen kann: $\mathcal{T}_1$ nimmt den Anfangsabschnitt u auf das Arbeitsband und versieht dabei die Trennzeichen $c_1, \dots, c_k$ jeweils mit Marken $m_1, \dots, m_k$. Beim Einlesen eines Blockes $v_p = 1^{i_{p,1}} c_1 1^{i_{p,2}} c_1 \dots c_1 1^{i_{p,k}} c_k x_p$ des Endabschnitts springt dann der Lesekopf zurück auf den Bandanfang (m Schritte, wenn die Marken m_k benutzt werden), von dort aus über $i_{p,1}-1$ Marken m_k zu den Blöcken mit $i_{p,1}$ als erstem Index, von dort aus über $i_{p,2}-1$ Marken m_{k-1} zu den Blöcken mit $i_{p,1}$ als erstem und $i_{p,2}$ als zweitem Index, ... , schließlich über $i_{p,k}-1$ Marken m_1 genau an den Anfang des Blockes $u_{i_{p,1}, \dots, i_{p,k}}$ im Anfangsabschnitt. Das Aufsuchen dieses Blockes und der Vergleich können damit in Linearzeit geschehen, wobei die Marken auf dem Anfangsabschnitt nicht mehr gelöscht oder versetzt werden,

sondern unverändert stehen bleiben. Es bleibt noch zu zeigen, wie für den Anfangsabschnitt eines Wortes w die Längenbedingungen (i) nachgeprüft werden können: Gleichzeitig mit dem Übertragen eines Blockes vom Eingabe- auf das Arbeitsband überprüft $\mathcal{T}_1$ durch Vergleich mit dem zuvor eingelesenen und schon abgespeicherten Block, ob auch der neue Block die Länge m hat. Dies kann durch Hin- und Herspringen, jeweils unter Verschieben einer beweglichen Marke m' geschehen, wie im folgenden Bild angedeutet:

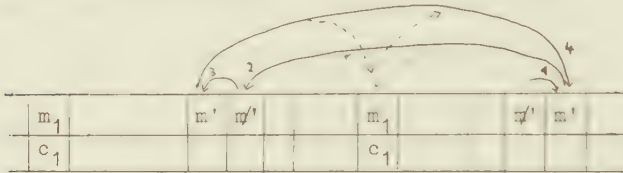


Abb.7: Vergleich von Blocklängen durch eine TAS $\mathcal{T}_1$

Es sind also jeweils 4. m zusätzliche Schritte nötig. In derselben Weise kann $\mathcal{T}_1$ überprüfen, ob die Bedingungen für die Folgen von jeweils $m, m^2, m^3, \dots$ Blöcken gelten, die jeweils durch die Zeichen $c_2, c_3, c_4, \dots$ voneinander getrennt sind. $\mathcal{T}_1$ springt, wenn ein Zeichen c_j ($2 \leq j \leq k+1$) gelesen wird, zum Längenvergleich jeweils auf die zuvor mit m_{j-1} markierten Zeichen c_{j-1} ; auch dafür reichen je 4. m zusätzliche Schritte aus ¹⁾. $\mathcal{T}_1$ kommt daher in Linearzeit mit

1) An zwei Stellen ist im Detail noch eine genauere Überlegung nötig:

a) Im Fall $k=1$ steht für $\mathcal{T}_1$ zum Vergleich der Blocklängen keine zusätzliche Marke m' mehr zur Verfügung, hier muß die Marke m_1 benutzt werden (6. m zusätzliche Schritte).

b) Beim Längenvergleich für die Folgen von Blöcken

(Fortsetzung S. 35)

einem Arbeitsband und k Marken aus.

Wie schon gesagt, ist es zur Überprüfung der Blöcke im Endabschnitt nicht nötig, daß die auf dem Anfangsabschnitt gesetzten Marken wieder gelöscht oder verschoben werden können. Für ein Überprüfen der Längenbedingungen im Anfangsabschnitt ist dagegen ein Hin- und Herspringen notwendig, wenn $\mathcal{T}_1$ nur k Marken besitzt. Mit einem zusätzlichen Arbeitsband, das ein Zählerband sein kann (d.h. ein Kellerband, dessen Bandalphabet nur aus einem Symbol besteht), kann $\mathcal{T}_1$ auch die Längenvergleiche ohne bewegliche Marken vornehmen (jeweils Schritte bzw. Sprünge auf dem Arbeitsband mit gleichzeitigem Abzählen von m auf dem Zählerband). Aber auch im Fall, daß $\mathcal{T}_1$ kein zusätzliches Band besitzt, kann die Bedingung, daß $\mathcal{T}_1$ Marken nur setzen, nicht aber löschen kann, eingehalten werden, wenn $\mathcal{T}_1$ noch zwei zusätzliche Marken m'_1, m'_2 hat: Die Überprüfung, ob zwischen je zwei Symbolen c_{j+1} genau m Teilabschnitte liegen, die jeweils durch ein Symbol c_j ($j \geq 1$) getrennt sind, kann von $\mathcal{T}_1$ in $C_7 \cdot m^2$ Schritten vorgenommen werden, wenn $\mathcal{T}_1$ nach dem üblichen Verfahren für 1-Band-Turingmaschinen die Felder mit c_j (die mit Hilfe der Marken m_j erreichbar sind) abzählt. Wegen $j+1 \geq 2$ wird damit die Bedingung, daß $\mathcal{T}_1$ in Linearzeit arbeitet, nicht verletzt. Es bleibt zu zeigen,

(Fortsetzung von S. 34)

zwischen den Zeichen c_j ($j \neq k-1$) kann jeweils die Marke m_{j+1} zum Hin- und Herspringen benutzt werden, für $j=k$ ist das jedoch nicht mehr möglich, da nur k Marken zur Verfügung stehen. Hier muß $\mathcal{T}_1$ jeweils die Marken m_k benutzen und braucht $C_6 \cdot m^2$ Schritte zum Überprüfen der Längenbedingungen.

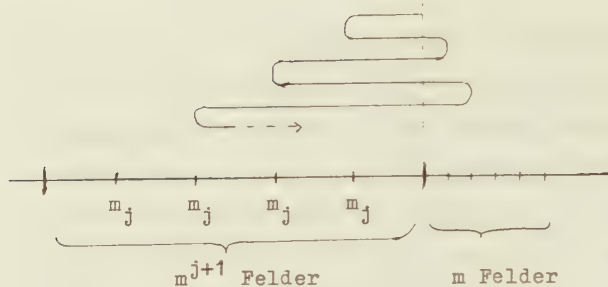


Abb. 9: Überprüfen der Längenbedingungen, $j \geq 1$

wie $\mathcal{T}_1$ Einzelblöcke der Länge m nach dem Einlesen vergleichen kann, ohne jeweils mehr als $C_8 \cdot m$ Schritte zu benötigen und ohne bewegliche Marken zu benutzen. Dazu nimmt $\mathcal{T}_1$ zunächst den ersten und zweiten Block der Eingabe auf das Arbeitsband und vergleicht dann deren Länge in der folgenden Weise: $\mathcal{T}_1$ markiert das letzte Feld von Block 2 mit der Marke m'_1 , läuft dann zum ersten Feld von Block 1 und markiert es ebenfalls mit m'_1 . Dann wird zu Block 2 zurückgesprungen, eine weitere Marke m'_1 auf das vorletzte Feld gesetzt und anschließend bei Block 1 eine Marke m'_1 auf das zweite Feld usw. Es werden also, von den Enden her beginnend, alle Felder beider Blöcke mit Marken m'_1 versehen, die dort stehenbleiben. Für den Längenvergleich von Block 2 und Block 3 wird in derselben Weise die Marke m'_2 verwendet, für Block 3 und Block 4 kann wieder m'_1 benutzt werden usw.

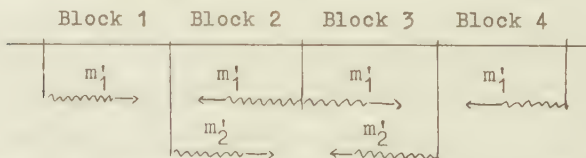


Abb. 10: Überprüfen der Längenbedingungen für Einzelblöcke

In diesem Fall kommt $\mathcal{T}_1$ also ganz ohne bewegliche Marken aus, (2) ist damit bewiesen.

Im Fall $k=1$ kann auch eine 1TAS den Erkennungsalgorithmus ausführen: Sie markiert sich die Zeichen c_1 auf dem Arbeitsband zunächst mit einer Marke m_1 , die sie dann beim Einlesen eines Blocks des Endabschnitts zum Aufsuchen des entsprechenden Blocks im Anfangsabschnitt benutzt. Dabei wird jede Marke m_1 durch m_2 ersetzt, beim nächsten Block des Endabschnitts kann also die Marke m_2 als Sprungziel dienen, dann wieder m_1 , usw. Auch das Überprüfen der Längenbedingungen für den Anfangsabschnitt kann mit Hilfe von zwei Marken in Linearzeit geschehen, (3) ist damit bewiesen.

Schließlich läßt sich leicht eine TM $\mathcal{T}_4$ mit $(k+1)$ -dimensionalem Speicherband angeben, die L_k akzeptiert. Die Arbeitsweise von $\mathcal{T}_4$ wird am besten durch ein Bild (für $k=2$, also $k+1=3$) deutlich, das angibt, wie $\mathcal{T}_4$ die Anfangsabschnitte von L_k speichert:

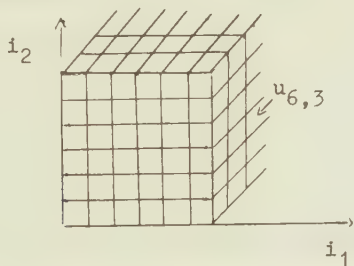


Abb.11: Speicher einer 3-dimensionalen TM für L_2

Die ersten Elemente der Blöcke $u_{i_1, \dots, i_k}$ ($1 \leq i_j \leq m$) sind also jeweils in Speicherzellen gespeichert, die zusammen eine k -dimensionale Hyperebene des $(k+1)$ -dimensionalen Speichers bilden. Wenn nun ein neuer Block des Endab-

schnittes mit dem Block $u_{i_1, \dots, i_k}$ zu vergleichen ist, so macht $\mathcal{J}_4$ zunächst für alle j jeweils $i_j - 1$ Schritte in Richtung der j -ten Dimension ($j = 1, \dots, k$) und hat dann beim Weitergehen in der $(k+1)$ -ten Dimension genau den gesuchten Block zur Verfügung. Da dies in Linearzeit möglich ist und auch, wie man sich leicht überlegt, das Überprüfen der Bedingungen für den Anfangsabschnitt in Linearzeit möglich ist, ist damit (4) bewiesen.

Durch Kombination von Satz 3.3 mit Lemma 3.1 und 3.2 lassen sich nun die gesuchten Aussagen über die relativen Fähigkeiten der verschiedenen Maschinenmodelle (jeweils als On-Line-Modelle) herleiten:

Korollar 3.4: Es gibt zu jedem $k \in \mathbb{N}$ ($k \geq 1$) eine Sprache

L_k , so daß gilt:

- (1) L_k kann nicht on-line in Linearzeit von einer TM mit k -dimensionalem Speicher akzeptiert werden, L_k kann jedoch on-line in Linearzeit von einer 1-Band-TAS $\mathcal{J}$ mit k Marken oder auch von einer TM mit $(k+1)$ -dimensionalem Speicher akzeptiert werden.
- (2) Verfügt die TAS $\mathcal{J}$ über zwei weitere Marken (insgesamt also $k+2$ Marken) oder über ein zusätzliches Zählerband (das auf Zahlen bis zur Größenordnung $k\sqrt{n}$ beschränkt sein kann), so kann für $\mathcal{J}$ als zusätzliche Bedingung angenommen werden, daß $\mathcal{J}$ Marken nur setzen, nicht aber löschen kann.

Korollar 3.5: Es gibt eine Sprache L_2 , so daß gilt: L_2

kann on-line in Linearzeit von einer TAS mit 2 Marken, nicht dagegen von einer 1TAS (mit beliebig, aber endlich

vielen Marken) oder von einer TAS mit nur einer Marke akzeptiert werden.

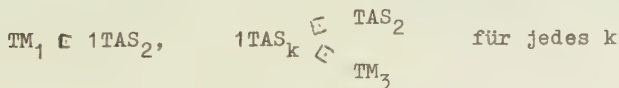
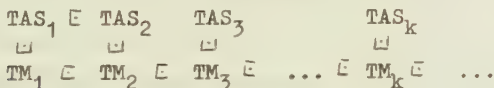
Korollar 3.6: Es gibt eine Sprache L_1 , so daß gilt: L_1 kann on-line in Linearzeit von einer 1TAS mit 2 Marken, nicht jedoch von einer (ein- oder mehrbändigen) TM mit eindimensionalem Speicherband akzeptiert werden.

Die Aussage über k - bzw. $(k+1)$ -dimensionale Turingmaschinen in Korollar 3.4 ist schon in [He 66] enthalten, wobei dort eine andere Sprache zum Beweis dient.

Aussage (2) von Korollar 3.4 wurde deshalb mit aufgenommen, weil Korollar 3.4 ein Gegenstück zu Satz 2.3 (S. 19) darstellt: Korollar 3.4 besagt, daß Satz 2.3 für das hier definierte Modell der TAS nicht gilt. Aussage (2) von Korollar 3.4 zeigt dabei, daß offenbar die Einschränkung (1) des Fischer/Rosenberg-Modells in Satz 2.3 die wesentliche ist, denn Korollar 3.4 gilt auch noch für ein Modell der TAS, das der Einschränkung (2) unterliegt.

Korollar 3.5 und 3.6 zeigen, daß das Modell der 1TAS in seinen Fähigkeiten echt zwischen dem der TM und dem der TAS liegt, und zwar unabhängig von der Markenzahl der 1TAS.

In anschaulicher Form lassen sich die Ergebnisse durch die beiden folgenden Diagramme darstellen:



Dabei bedeutet " $A \sqsubseteq B$ ": Es gibt eine Sprache L , die von einer On-Line-Maschine vom Typ B in Linearzeit erkannt werden kann, dagegen nicht in Linearzeit von einer On-Line-Maschine vom Typ A . TAS_k ($1TAS_k$) bezeichnet den Typ der TAS bzw. $1TAS$ mit k Marken (ein Arbeitsband und ein Eingabeband) und TM_k den Typ der TM mit k -dimensionalem Speicher (mit beliebiger endlicher Anzahl von Bändern).

Offen bleibt, wie in diesem Sinne das gegenseitige Verhältnis von $1TAS_1$ und TAS_1 sowie von TM_2 und $1TAS_k$ ist. Da alle diese Maschinentypen Speicherzugriff von der Ordnung 2 haben, läßt sich zur Beantwortung dieser Frage Satz 3.3 nicht anwenden. Ebenso bleibt vorläufig offen, ob $TAS_k \sqsubseteq TAS_{k+1}$ auch für allgemeines $k > 1$ gilt. Um hierfür Satz 3.3 anwenden zu können, müßte gezeigt werden, daß eine TAS mit k Marken Speicherzugriff von der Ordnung $k+1$ hat. Es ist zu vermuten, daß das gilt; ein Beweis müßte aber berücksichtigen, daß hier k Marken in beliebig kompliziert verschachtelter Anordnung, nicht nur in hierarchischer Anordnung wie im Beweis zu Satz 3.3, auftreten können.

Es sollte noch darauf hingewiesen werden, daß die Schreibweise " $A \sqsubseteq B$ " nicht notwendig etwas über die Simulierbarkeit einer Maschine vom Typ A durch eine Maschine vom Typ B besagt. Diese Frage soll im nächsten Abschnitt behandelt werden, dabei werden allerdings nicht Zusammenhänge zwischen TAS und TM betrachtet, sondern die Abhängigkeit der Zeitbeschränkung bei der TAS von der Anzahl der Marken.

4. Markenzahl und Komplexität bei der TAS

Ähnlich wie bei der TM ein Zusammenhang zwischen Anzahl der Bänder und Zeitkomplexität besteht, ist ein solcher Zusammenhang bei der TAS zwischen Anzahl der Marken und Zeitkomplexität zu vermuten; er soll im folgenden Abschnitt behandelt werden. Es soll untersucht werden, wie stark die Zeitbeschränkung vergrößert werden muß, damit eine TAS mit einer größeren Anzahl von Marken durch eine TAS mit einer geringeren Anzahl von Marken simuliert werden kann. Eine solche Simulation ist möglich, wenn gleichzeitig von einer festen Bandbeschränkung $L(n)$ ausgegangen werden kann. Diese wiederum läßt sich, wenn sie nicht explizit gegeben ist, auf alle Fälle durch die Zeitbeschränkung $T(n)$ abschätzen, da wie eine TM auch eine TAS in $T(n)$ Schritten höchstens einen Abschnitt von $T(n)$ Bandfeldern benutzen kann. Zunächst soll eine solche Beziehung für $L(n) = n^k$ bewiesen werden, da hier die verwendete Beweisidee am naheliegendsten ist; später wird die Aussage für beliebige Bandbeschränkung $L(n)$ verallgemeinert. Im folgenden soll dabei immer angenommen werden, daß Bandbeschränkung $L(n)$ und Zeitbeschränkung $T(n)$ nicht nur beliebige obere Abschätzungen für Bandbedarf und Schrittzahl angeben, sondern daß immer $\lim_{n \rightarrow \infty} \frac{T(n)}{L(n)} > 0$ gilt. Da für die folgenden Sätze das Eingabeband keine Rolle spielt, soll für die weiteren Abschnitte ein Off-Line-Modell der TAS angenommen werden. Das Eingabeband entfällt also, die Eingabe steht zu Beginn der Rechnung auf dem Arbeitsband.

Satz 4.1: Sei $k \in \mathbb{N}$ fest, $\mathcal{T}$ eine 1-Band-TAS mit Bandbeschränkung $L(n) = n^k$ und Zeitbeschränkung $T(n)$, $\mathcal{T}$ habe beliebig, jedoch endlich viele Marken. Dann gibt es eine zu $\mathcal{T}$ äquivalente 1-Band-TAS $\mathcal{T}'$ mit k Marken, die mit Zeitbeschränkung $(4k+1) \cdot n \cdot T(n)$ und Bandbeschränkung n^k arbeitet.

Beweis: Der Grundgedanke des folgenden Beweises ist, daß $\mathcal{T}'$ zur Simulation eines Sprunges von $\mathcal{T}$ auf eine Marke sich über eine Hierarchie von $k-1$ Marken, die bei einem Eingabewort der Länge n jeweils n^i ($1 \leq i \leq k-1$) Felder voneinander entfernt stehen, an das Sprungziel "herantastet". Dazu ist das Band von $\mathcal{T}'$ in der folgenden Weise eingeteilt: Nach je n Feldern ist ein zusätzliches sogenanntes Referenzfeld eingeschoben, das als Zwischen-Sprungziel dient; Marken von $\mathcal{T}'$ werden nur auf diese Felder gesetzt. Die Anzahl der Bandsymbole von $\mathcal{T}'$ hängt außer vom Bandalphabet von $\mathcal{T}$ auch von der Anzahl der Marken von $\mathcal{T}$ ab: Wenn $\mathcal{T}$ k' Marken hat, ist die Symbolspur von $\mathcal{T}'$ in $k'+1$ Spuren unterteilt, eine für die Bandsymbole von $\mathcal{T}$ und je eine für die Marken $m_1, \dots, m_{k'}$ von $\mathcal{T}$ bzw. die zusammengesetzten Symbole $\binom{m_1}{R}, \dots, \binom{m_{k'}}{R}, \binom{m_1}{L}, \dots, \binom{m_{k'}}{L}$, deren Funktion später erläutert wird. Die Felder von $\mathcal{T}'$, die nicht Referenzfelder sind, werden im Verlauf der Berechnung in derselben Weise beschrieben, wie dies $\mathcal{T}$ mit den entsprechenden Feldern auf ihrem Band tut, wobei jedoch die Marken $m_1, \dots, m_{k'}$ in der Symbolspur von $\mathcal{T}'$ wie gewöhnliche (Teil-)Symbole gesetzt oder gelöscht werden. Die Referenzfelder auf dem Band von $\mathcal{T}'$ sind mit den Marken $m'_1, \dots, m'_{k-1}$

in der folgenden Weise markiert: Referenzfelder, die jeweils n^i Felder ($1 \leq i \leq k-1$) voneinander entfernt sind, werden als Referenzfelder i -ter Stufe bezeichnet und mit allen Marken $m'_1, \dots, m'_i$ markiert. Um einen Bandabschnitt von $n^k + n^{k-1} + 1$ Feldern (Bandbedarf von $\mathcal{T}$ zuzüglich der Referenzfelder) in dieser Weise zu überdecken, kommt $\mathcal{T}'$ mit $k-1$ Marken aus. Wie $\mathcal{T}'$ sich jeweils Bandabschnitte der Länge n^i ($1 \leq i \leq k-1$) in dieser Weise einrichtet, wird später behandelt. In der Symbolspur speichert sich $\mathcal{T}'$ in den Referenzfeldern jeweils das Symbol $\binom{m}{L}$ bzw. $\binom{m}{R}$, wenn in den entsprechenden Bandabschnitten bei $\mathcal{T}$ zwischen diesem Feld und dem nächsten Referenzfeld der gleichen Stufe nach links bzw. rechts die Marke m (die für $\mathcal{T}'$ ein normales (Teil-)Bandsymbol ist) vorkommt; ggf. wird $\binom{m}{LR}$ gespeichert (m kommt links und rechts vor). Diese Organisation soll durch das folgende Bild verdeutlicht werden:

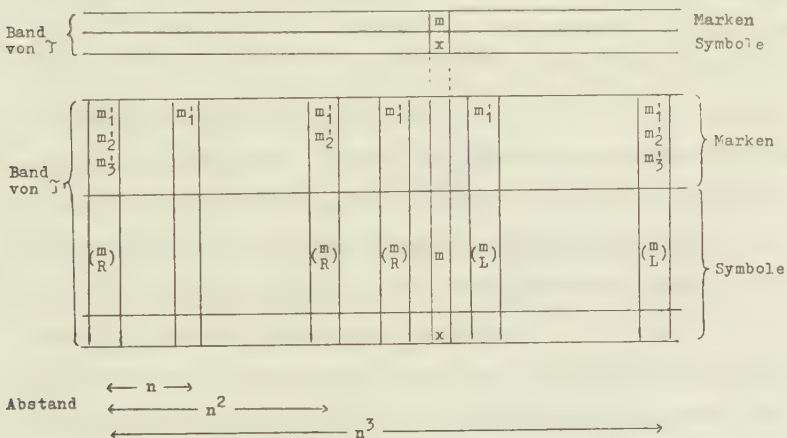


Abb.12: Bandorganisation von $\mathcal{T}'$ für $k \geq 3$

$\mathcal{T}'$ speichert also in den Referenzfeldern jeweils Verweise darauf, wo bei $\mathcal{T}$ jeweils Marken m vorkommen; die Marken m_i^1 ($i \leq k-1$) dienen zum schnellen Auffinden der Referenzfelder, m_k^1 zur Kennzeichnung des Arbeitsfeldes von $\mathcal{T}$.

Nun ist zu zeigen,

- (1) wie von $\mathcal{T}'$ ein Sprung von $\mathcal{T}$ auf eine Marke simuliert wird,
- (2) wie $\mathcal{T}'$ die Organisation auf den Referenzfeldern zu verändern hat, wenn $\mathcal{T}$ eine Marke neu setzt oder löscht,
- (3) wie die hier beschriebene Bandorganisation hergestellt werden kann

und wieviel Schritte dazu jeweils nötig sind.

Zu (1): Sei o.B.d.A. ein Sprung von $\mathcal{T}$ auf die Marke m nach links zu simulieren. $\mathcal{T}'$ läuft dann in Einzelschritten nach links bis zum nächsten Referenzfeld der Stufe 1. Wenn nicht schon auf dem Weg dorthin die Marke m gefunden wird, so wird im Referenzfeld nachgesehen, ob ein Symbol $(\frac{m}{L})$ in der entsprechenden Spur steht. Wenn ja, läuft $\mathcal{T}'$ in Einzelschritten weiter, wenn nein, dann springt die Maschine zum nächsten Referenzfeld der Stufe 1 usw. Allgemein verfährt $\mathcal{T}'$ dabei in der folgenden Weise: Wird zum ersten Mal ein Feld einer höheren Referenzstufe erreicht, so erfolgt, wenn sich dort kein Symbol $(\frac{m}{L})$ befindet, ein Sprung bis zum nächsten Referenzfeld derselben (höheren) Stufe, denn dann ist sicher, daß sich keine Marke m auf den Feldern dazwischen befindet. Auf diese Weise fährt $\mathcal{T}'$ fort, bis auf einem Referenzfeld der Stufe i ($i \leq k-1$) ein Symbol $(\frac{m}{L})$ gefunden wird. Dann wird zu den Referenzfeldern der

nächstniedrigeren Stufe gesprungen, bis dort ein Symbol $(\frac{m}{L})$ erscheint, von dort auf der nächstniedrigeren Stufe bis zum nächsten Symbol $(\frac{m}{L})$ usw. bis schließlich in Einzelschritten die Marke m erreicht ist. Es kann dabei auch vorkommen, daß nach dem ersten Sprung von einem Referenzfeld einer neuen Stufe aus statt eines Symbols $(\frac{m}{L})$ ein Symbol $(\frac{m}{R})$ gefunden wird - wenn nämlich die gesuchte Marke von $\mathcal{T}$ hier schon übersprungen wurde -; in diesem Fall muß die weitere Suche wieder von rechts her erfolgen. Der Suchprozeß nach einer Marke m kann für ein Beispiel durch das folgende Bild verdeutlicht werden:

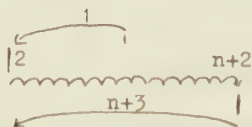


Abb.13: Suche nach einer Marke m

Durch die diesem Suchverfahren angepaßte Bandorganisation benötigt $\mathcal{T}'$ zum Auffinden einer Marke m (genauer: eines Teilsymbols, das einer Marke m von $\mathcal{T}$ entspricht) auf jeder Referenzstufe höchstens $2 \cdot n$ Schritte bzw. Sprünge, insgesamt also höchstens $2 \cdot n \cdot k$ Schritte.

Zu (2): Ist von $\mathcal{T}'$ das Setzen einer oder mehrerer Marken von $\mathcal{T}$ zu simulieren, so sind auf allen "benachbarten" Referenzfeldern aller Stufen die entsprechenden Symbole

$(\frac{m}{L})$ bzw. $(\frac{m}{R})$ einzutragen, und zwar in beiden Richtungen. Ist außerdem auch das Löschen einer bzw. mehrerer Marken von $\mathcal{T}$ zu simulieren, so muß jeweils überprüft werden, ob die jeweilige Marke m die einzige ihres Typs in dem betreffenden Abschnitt zwischen zwei Referenzfeldern war - in diesem Fall sind die entsprechenden Symbole $(\frac{m}{L})$ bzw. $(\frac{m}{R})$ zu löschen - oder ob an anderer Stelle eine weitere Marke m vorkommt - dann müssen die Symbole $(\frac{m}{L})$ bzw. $(\frac{m}{R})$ stehenbleiben. Es sind also jeweils alle Referenzfelder der Stufe $i-1$ zu prüfen, bevor eine Entscheidung über das Löschen eines Symbols $(\frac{m}{L})$ bzw. $(\frac{m}{R})$ in einem Referenzfeld der Stufe i getroffen werden kann. Dies kann jedoch für alle Marken von $\mathcal{T}$ gleichzeitig geschehen (die Information läßt sich im endlichen Gedächtnis von $\mathcal{T}'$ speichern), außerdem kann das Setzen neuer Marken im selben Arbeitsgang simuliert werden. Für jede Referenzstufe sind beim Löschen einer Marke von $\mathcal{T}$ $n+3$ Schritte nötig und aus-



Suchen
evtl. Löschen, Setzen
auf den Referenzfeldern

Abb.14: Löschen von Marken

reichend, insgesamt also für das Setzen und Löschen von Marken bei einem Schritt von $\mathcal{T}$ $k.(n+3)$ Schritte von $\mathcal{T}'$.

Zu (3): Zu Beginn ist durch das Eingabewort ein Block von n Feldern gegeben. Wird dieser Block zum ersten Mal verlassen, so werden weitere $n-1$ Blöcke mit der oben beschriebenen Bandorganisation vorbereitet (jeweils n Felder, dazu

die Referenzfelder, zusammen also $(n-1)(n+1)$ weitere Felder). Allgemein schiebt $\mathcal{T}'$, wenn ein bisher vorbereiteter (d.h. mit Referenzfeldern versehener) Abschnitt des Arbeitsbandes zwischen zwei Referenzfeldern der Stufe i - ein solcher Abschnitt hat die Länge $n^{i-1}(n+1) + 1$ - erstmalig verlassen wird, eine Anzahl von Schritten ein, in denen weitere Referenzfelder eingerichtet werden. Es wird dann durch $(n-1)$ -maliges Übertragen der bisher vorhandenen Bandorganisation ein weiterer Bandbereich der Länge $(n-1)n^{i-1}(n+1)$, zusammen mit dem bisherigen also $n^i(n+1) + 1$ Felder, mit Referenzfeldern versehen, so daß anschließend auf diesem Feldern in der in (1) und (2) beschriebenen Weise gearbeitet werden kann. Dazu sind jeweils $4 \cdot n^i \cdot (n+1) \cdot (n-1) \leq 4 \cdot n^{i+2}$ Schritte zum Kopieren und $3 \cdot n$ Schritte zum Mitzählen nötig ¹⁾. Bei der letzten Stufe legt $\mathcal{T}'$ das Band nach beiden Richtungen mit Referenzfeldern der Stufe $k-1$ aus, so daß in beiden möglicherweise nötigen Richtungen n^k Felder zur Verfügung stehen. Insgesamt benötigt $\mathcal{T}'$ damit höchstens

$$\begin{aligned} & 4 \cdot ((n+1)(n-1) + (n+1)n(n-1) + (n+1)n^2(n-1) + \dots \\ & \quad \dots + (n+1)n^{k-3}(n-1) + 2 \cdot (n+1)n^{k-2}(n-1)) + 2 \cdot n \cdot k \\ & = 4 \cdot (n^2 - 1) \cdot (1 + \dots + n^{k-3} + 2 \cdot n^{k-2}) + 3 \cdot n \cdot k \\ & \leq 9 \cdot n^k \end{aligned}$$

Schritte und eine zusätzliche Marke m_k^1 .

Zur Simulation eines Schrittes von $\mathcal{T}$ müssen gleichzeitig (1) und (2) beachtet werden, da in einem Schritt von $\mathcal{T}$ ja

1) Allgemein gilt für die TAS, daß in $4 \cdot p$ Schritten eine Zeichenkette der Länge p von einer Stelle des Bandes an eine andere übertragen werden kann, wenn die TAS zum Hin- und Herspringen eine sonst nicht benutzte Marke (hier die Marke m_k^1) zur Verfügung hat; vgl. Abb.7, S.34.

gleichzeitig das Setzen oder Löschen von Marken und ein Sprung auf eine andere Marke geschehen kann. Jeder Schritt von $\mathcal{T}$ kann nach (1) und (2) durch $2.k.n + k.(n+3) \leq 4.k.n$ (für genügend großes n) Schritte von $\mathcal{T}'$ simuliert werden, dazu kommen noch die insgesamt höchstens $9.n^k$ Schritte zum Herstellen der Bandorganisation von $\mathcal{T}'$. Wegen $\lim_{n \rightarrow \infty} \frac{T(n)}{n^k} > 0$ gilt $n.T(n) > 9.n^k$ für genügend große n , die Zeitbeschränkung von $\mathcal{T}'$ läßt sich daher durch $(4k+1).n.T(n)$, die Bandbeschränkung durch $2.n^{k-1}(n+1) + 1$ abschätzen, wegen des Speed-Up-Satzes für die Bandbeschränkung (Lemma 2.1) kann die Bandbeschränkung auch für $\mathcal{T}'$ auf n^k reduziert werden. $\mathcal{T}'$ kommt außerdem mit k Marken $m_1^1, \dots, m_k^1$ aus, Satz 4.1 ist damit bewiesen.

Im Beweis zu Satz 4.1 wurde wiederum, ähnlich wie bei Satz 3.3, die Idee der indexsequentiellen Speicherung auf die TAS angewandt. Dabei ist eine k -stufige Hierarchie von Referenzfeldern bei einer Bandbeschränkung $L(n) = n^k$ besonders naheliegend. Es wäre jedoch z.B. auch möglich, mit Hilfe einer $2.k$ -stufigen Hierarchie eine Simulation von $\mathcal{T}$ durch $\mathcal{T}'$ mit Zeitbeschränkung $C.2.k.\sqrt[n]{n}.T(n)$ zu erreichen, oder mit $4.k$ Stufen $C.4.k.\sqrt[4]{n}.T(n)$ usw. Der folgende Satz 4.4 erweitert die Aussage von Satz 4.1 für beliebige Bandbeschränkung und beliebige Markenzahl. Gleichzeitig wird die simulierende Maschine $\mathcal{T}'$ näher charakterisiert als eine TAS in "Normalform"; diese genauere Charakterisierung wird später noch benötigt.

Definition: Eine TAS $\mathcal{T}$ heißt TAS in Normalform mit Parametern k, s und f , wenn sie die folgenden Bedingungen

erfüllt:

- (1) $\mathcal{T}$ hat k Marken $m_1, \dots, m_k$,
- (2) das Bandalphabet von $\mathcal{T}$ hat s Symbole,
- (3) alle Marken bis auf eine, o.B.d.A. die Marken $m_1, \dots, m_{k-1}$, werden, wenn sie einmal gesetzt sind, nicht wieder gelöscht ("feste Marken"). Für das Setzen einer festen Marke gilt außerdem ¹⁾: Eine solche Marke m_j wird nie zwischen zwei schon gesetzte Marken des gleichen Typs gesetzt, und ein Sprung auf eine solche Marke in einer Richtung des Bandes wird nur ausgeführt, wenn in der anderen Richtung diese Marke ebenfalls vorhanden ist. (Es werden also immer "am Rand" neue Marken gesetzt, und es wird nur "innerhalb" markierter Bereiche gesprungen.)
- (4) Die Marke m_k ("bewegliche Marke") steht zu jedem Zeitpunkt einer Berechnung auf höchstens f Feldern des Bandes ²⁾.

Zum Beweis von Satz 4.4 werden noch die Definition der TAS-berechenbaren und der TAS-gutartigen Funktion sowie zwei Lemmata benötigt.

Definition: Eine Abbildung $f: \mathbb{N} \rightarrow \mathbb{N}$ wird von einer TAS mit Bandbeschränkung $L(n)$ und Zeitbeschränkung $T(n)$ be-

-
- 1) Der Grund für die folgende Einschränkung wird in einem späteren Beweis (Satz 6.2) deutlich.
 - 2) Wie man sich leicht überlegt, stellt die Bedingung, daß es nur eine solche Marke gibt, keine wesentliche Einschränkung dar: Eine TAS mit p beweglichen Marken, von denen jede nur auf höchstens f Feldern des Bandes gleichzeitig stehen kann, kann in Linearzeit von einer TAS mit nur einer beweglichen Marke, die auf höchstens $f \cdot p$ Feldern gleichzeitig stehen kann, simuliert werden.

rechnet, wenn $\tilde{T}$ bei Eingabe von 1^n auf dem Arbeitsband die unäre Darstellung von $f(n)$ berechnet und dazu nicht mehr als $L(n)$ Bandfelder und $T(n)$ Schritte benötigt. f heißt TAS-gutartig, wenn es eine TAS $\tilde{T}$ mit nicht mehr als zwei Marken und eine Konstante C gibt, so daß f von $\tilde{T}$ mit Zeitbeschränkung $f(n)$ berechnet wird. Ein Funktionspaar $(L(n), T(n))$ heißt TAS-gutartig, wenn es eine TAS $\tilde{T}$ mit nicht mehr als zwei Marken und Konstanten C_1, C_2 gibt, so daß sowohl $L(n)$ als auch $T(n)$ von $\tilde{T}$ mit Bandbeschränkung $C_1 \cdot L(n)$ und Zeitbeschränkung $C_2 \cdot T(n)$ berechnet wird. Die Eigenschaft, TAS-gutartig zu sein, entspricht der Eigenschaft bei Turingmaschinen, "countable" zu sein, sie wird üblicherweise für Komplexitätsfunktionen vorausgesetzt (s. etwa [HU 69], S. 152/153).

Lemma 4.2: Sind zwei Zahlen p, q ($p \leq q$) in unärer Darstellung gegeben, so kann eine 1-Band-TAS in Normalform mit Parametern $k=1, s=3, f=3$ in $4 \cdot p \cdot q + 3 \cdot p$ Schritten die unäre Darstellung des Produkts $p \cdot q$ berechnen.

Beweis: Die Multiplikation wird als wiederholte Addition ausgeführt, jede Addition einer unär gespeicherten Zahl q ist als Übertragung einer Zeichenkette der Länge q in $4 \cdot q$ Schritten (s. Anmerkung im Beweis zu Satz 4.1, S. 47) möglich, dazu kommen noch $3 \cdot p$ Schritte für das Mitzählen der ausgeführten Additionen. Man überlegt sich leicht, daß man mit 3 Bandsymbolen und einer (beweglichen) Marke auskommt, wobei die Marke zu jedem Zeitpunkt höchstens dreimal (zweimal für die Addition, einmal zum Mitzählen) auf dem Band auftreten muß.

Lemma 4.3: Sei $p \geq 2$, $m \geq 4$. Dann ist die Berechnung von $\lceil \sqrt[p]{m} \rceil$ ¹⁾ in unärer Darstellung für eine TAS in Normalform mit Parametern $k=1$, $s=3$, $f=5$ in $9 \cdot (\lceil \sqrt[p]{m} \rceil)^{p+1}$ Schritten möglich. Insbesondere ist die Funktion $f(n) = n^{1 + \frac{1}{p}}$ TAS-gutartig.

Beweis: Ist eine Zahl r in unärer Darstellung gegeben, so läßt sich nach Lemma 4.2 nacheinander r^2 in $4 \cdot r^2 + 3 \cdot r$, r^3 in $4 \cdot r^3 + 3 \cdot r$, ..., r^p in $4 \cdot r^p + 3 \cdot r$ Schritten berechnen, dazu kommen $3 \cdot p$ Schritte zum Mitzählen von p . Insgesamt sind dies $4 \cdot (r^2 + r^3 + \dots + r^p) + 3 \cdot p \cdot (r+1)$ Schritte, also wegen $r + r^2 + r^3 + \dots + r^p = \frac{r^{p+1} - 1}{r - 1} \leq 2 \cdot r^p$ höchstens $8 \cdot r^p + 3 \cdot p \cdot (r+1)$ Schritte, die zur Berechnung von r^p benötigt werden. Ist für gegebenes m $\lceil \sqrt[p]{m} \rceil$ zu berechnen, so wird nacheinander $1^p, 2^p, 3^p, \dots, q^p$ berechnet, bis $q^p \geq m$ wird. Dazu benötigt eine TAS höchstens $8 \cdot (1^p + \dots + q^p) + 3 \cdot p \cdot (2+3+\dots+q+1)$, wegen $1^p + \dots + q^p \leq q^{p+1}$ also höchstens $8 \cdot q^{p+1} + 2 \cdot p \cdot q^2 \leq 9 \cdot q^{p+1} = 9 \cdot (\lceil \sqrt[p]{m} \rceil)^{p+1}$ Schritte. Ähnlich wie bei Lemma 4.2 überlegt man sich, daß es zum Mitzählen der Größen k, r, s und zur Multiplikation ausreicht, wenn eine (bewegliche) Marke bis zu fünfmal (dreimal für die Multiplikationen, je einmal zum Mitzählen von p und q) gleichzeitig auf dem Band vorkommt.

Satz 4.4: Sei $\tilde{T}$ eine 1-Band-TAS mit Bandbeschränkung $L(n)$ und Zeitbeschränkung $T(n)$, $\tilde{T}$ habe k' Marken und s' Bandsymbole, das Funktionspaar $(L(n), T(n))$ sei TAS-gutartig. Dann gibt es zu jeder natürlichen Zahl k , $k \geq 2$, eine zu $\tilde{T}$ äquivalente 1-Band-TAS $\tilde{T}'$ in Normalform mit

1) $\lceil z \rceil$ bezeichnet die kleinste ganze Zahl $\geq z$.

Parametern k (also k Marken), $s = s' \cdot 5^{k'}$ und $f = 5$, die mit Zeitbeschränkung

$$(4 \cdot k + C) \cdot k \sqrt{L(n)} \cdot T(n) \quad 1)$$

bzw. (wegen $L(n) \leq T(n)$)

$$(4 \cdot k + C) \cdot T(n)^{1 + \frac{1}{k}}$$

und Bandbeschränkung $L(n)$ arbeitet.

Beweis: Der Grundgedanke des Beweises zu Satz 4.1 kann beibehalten werden: $\tilde{T}$ richtet auf einem genügend großen Bandabschnitt eine k -stufige Hierarchie von Referenzfeldern ein und simuliert dann jeden Schritt von $\tilde{T}$ in $4 \cdot k \cdot n_{L,k}$ Schritten, wobei für $n_{L,k}$, das nunmehr im Beweis die frühere Rolle von n spielt, gilt: $n_{L,k} := \lceil k \sqrt{L(n)} \rceil$. Damit kann wegen $n_{L,k}^k \geq L(n)$ durch eine Hierarchie von $k-1$ Marken wie bei Satz 4.1 der gesamte von $\tilde{T}$ benutzte Bandabschnitt überdeckt werden; Punkt (1) und (2) des Beweises zu Satz 4.1 können also übernommen werden. Dabei kann zusätzlich für $\tilde{T}$ festgestellt werden: Als Symbolalphabet ist das kartesische Produkt aus dem Symbolalphabet von $\tilde{T}$ und $\{\text{leer}, m, \binom{m}{L}, \binom{m}{R}, \binom{m}{LR}\}^{k'}$ ausreichend, $\tilde{T}$ hat also $s' \cdot 5^{k'}$ Bandsymbole. Die von $\tilde{T}$ auf den Referenzfeldern zu setzenden Marken werden, wenn sie einmal gesetzt sind, nicht mehr gelöscht, und auch die Reihenfolge der Eintragung dieser Marken entspricht der Definition einer TAS in Normalform.

1) Hier wird, um die wesentliche Aussage des Satzes deutlicher auszudrücken, die Aussage "es gibt eine TAS $\tilde{T}$ ", die ... mit Zeitbeschränkung $(4 \cdot k + C) \cdot k \sqrt{L(n)} \cdot T(n)$ arbeitet" verwendet an Stelle der eigentlich korrekten Ausdrucksweise "es gibt eine Konstante C und eine TAS $\tilde{T}$ ", die ... mit Zeitbeschränkung $(4 \cdot k + C) \cdot k \sqrt{L(n)} \cdot T(n)$ arbeitet". Entsprechend sind auch die folgenden Sätze der Arbeit aufzufassen, soweit es um die Verwendung von Konstanten geht.

Zur Simulation der jeweiligen Kopfstellung von $\tilde{J}$ durch $\tilde{J}'$ genügt eine bewegliche Marke, sie braucht zu jedem Zeitpunkt nicht mehr als höchstens viermal auf dem Band vorhanden zu sein.

Damit ist nur noch zu zeigen, daß die Bandorganisation analog zu Punkt (3) des Beweises zu Satz 4.1 von $\tilde{J}'$ hergestellt werden kann.

Zum Herstellen dieser Bandorganisation markiert $\tilde{J}'$ für ein gegebenes Wort der Länge n zunächst $L(n)$ Felder auf dem Band, dafür reichen 2 Marken, $C_1 \cdot T(n)$ Schritte und $C_2 \cdot L(n)$ Bandfelder (C_1, C_2 geeignete Konstanten) aus. Dann wird $n_{L,k} := \lceil k\sqrt{L(n)} \rceil$ berechnet, nach Lemma 4.3 reichen dafür $9 \cdot n_{L,k}^{k+1}$ Schritte aus. Ein Abschnitt von $2 \cdot L(n) + 2 \cdot \frac{L(n)}{n_{L,k}}$ Feldern (d.h. $L(n)$ Arbeitsfelder und $\lceil \frac{L(n)}{n_{L,k}} \rceil$ Referenzfelder in jeder Richtung) wird dann mit der im Beweis zu Satz 4.1 beschriebenen Bandorganisation versehen, dies geschieht wie in Punkt (3) des Beweises zu Satz 4.1 skizziert; an die Stelle von n tritt hier jedoch $n_{L,k}$. $\tilde{J}'$ benötigt dazu für genügend großes n höchstens $9 \cdot n_{L,k}^k$ Schritte. Die Gesamtzahl der für den Aufbau der Bandorganisation nötigen Schritte beträgt also höchstens $9 \cdot n_{L,k}^{k+1} + 9 \cdot n_{L,k}^k$. Nach Konstruktion von $n_{L,k}$ gilt $n_{L,k} \leq k\sqrt{L(n)} + 1$, also $n_{L,k}^{k+1} \leq (k\sqrt{L(n)} + 1)^k \cdot (k\sqrt{L(n)} + 1) \leq 2 \cdot L(n) \cdot k\sqrt{L(n)}$ für genügend große n . Damit ist die Anzahl von Schritten zum Einrichten der Bandorganisation von $\tilde{J}'$ für genügend große n durch $9 \cdot n_{L,k}^{k+1} + 9 \cdot n_{L,k}^k \leq 10 \cdot n_{L,k}^{k+1} \leq 20 \cdot L(n) \cdot k\sqrt{L(n)}$ beschränkt. Für die eigentliche Simulation von $\tilde{J}$ durch $\tilde{J}'$ gilt analog zum Beweis von Satz 4.1, daß ein Schritt von $\tilde{J}$ durch

$4.k.n_{L,k}$ Schritte von $\tilde{J}$ simuliert werden kann, für genügend große n läßt sich dies wegen $n_{L,k} \approx \sqrt[k]{L(n)} + 1$ durch $(4.k+1) \cdot \sqrt[k]{L(n)}$ abschätzen. Insgesamt arbeitet damit $\tilde{J}$ mit Zeitbeschränkung

$$20 \cdot \sqrt[k]{L(n)} \cdot L(n) + (4.k+1) \cdot \sqrt[k]{L(n)} \cdot T(n) \\ \leq (4.k+21) \cdot \sqrt[k]{L(n)} \cdot T(n).$$

Wird $L(n)$ nicht explizit in die Rechnung einbezogen, sondern überall durch $T(n)$ abgeschätzt, so ergibt sich als Zeitbeschränkung

$$(4.k+21) \cdot T(n)^{1 + \frac{1}{k}}.$$

Die Bandbeschränkung $(2.L(n) + 2 \cdot \left\lceil \frac{L(n)}{n_{L,k}} \right\rceil + L_1(n))$ läßt sich wegen Lemma 2.1 auf $L(n)$ reduzieren; Satz 4.4 ist damit bewiesen.

Die Frage liegt nahe, für welche Kombination von $L(n)$ und k die Simulation optimal ist, d.h. der zu $T(n)$ tretende Faktor minimal ist. Dazu ist, wenn für festes n $L := L(n)$ gesetzt wird, $(k+C) \cdot \sqrt[k]{L}$ als Funktion von k zu betrachten. Nach den Regeln der Differentialrechnung gilt für $(x+C) \cdot L^{\frac{1}{x}}$ als Funktion von x : Aus $\frac{d}{dx} (x+C) \cdot L^{\frac{1}{x}} = 0$ folgt

$$x = \frac{\log L}{2} \pm \frac{1}{2} \sqrt{(\log L)^2 + 4.C \cdot \log L}$$

(hier $\log$ = natürlicher Logarithmus).

Für $x = \log L \cdot (\frac{1}{2} + \frac{1}{2} \sqrt{1 + \frac{4.C}{\log L}})$ ist, wie man ebenso nach-

rechnet, $\frac{d^2}{dx^2} ((x+C) \cdot L^{\frac{1}{x}}) > 0$, also hat $(x+C) \cdot L^{\frac{1}{x}}$ dort ein Minimum. Der Faktor $(k+C) \cdot L^{\frac{1}{k}}$ wird also für

$$k \approx \log L \cdot (\frac{1}{2} + \frac{1}{2} \sqrt{1 + \frac{4.C}{\log L}})$$

minimal; mit wachsendem L geht $\frac{4.C}{\log L}$ gegen 0, und man erhält $k = \log L$. Vernachlässigt man also (für genügend große L bzw. k ist das gerechtfertigt) die Konstante C , so ist der zu $T(n)$ tretende Faktor $k.L^{\frac{1}{k}} = \log L \cdot e^{\frac{1}{\log L} \cdot \log L} = e \cdot \log L$, der zu $L(n)$ tretende Faktor ist $L^{\frac{1}{k}} = L^{\frac{1}{\log L}} = e$. $\mathcal{T}'$ arbeitet also bei der Simulation von $\mathcal{T}$ im günstigsten Fall mindestens mit der $(e \cdot \log L)$ -fachen Zeitkomplexität von $\mathcal{T}$. Selbstverständlich gilt dieser günstigste Fall mit der Beziehung $k = \log L$ nur für ein festes n (L ist ja eine Funktion von n) und ist damit für die Simulation einer TAS durch eine andere TAS, deren Markenzahl vorweg allgemein festzulegen ist und die Eingabeworte einer beliebigen Länge n verarbeiten soll, weniger interessant; in einem anderen Zusammenhang (Simulation der TAS durch eine RAM) wird aber später auf diese Überlegung noch zurückgegriffen werden.

Schließlich läßt sich durch Heranziehen von Lemma 2.2 Satz 4.4 erweitern auf eine TAS mit mehreren Bändern:

Korollar 4.5: Sei $\mathcal{T}$ ein r -bändige TAS mit k' Marken, die mit Bandbeschränkung $L(n)$ und mit Zeitbeschränkung $T(n)$ arbeitet. Dann gibt es zu jeder natürlichen Zahl k , $k \geq 2$, eine zu $\mathcal{T}$ äquivalente 1-Band-TAS $\mathcal{T}'$ mit k Marken, die mit Zeitbeschränkung

$$(16.r.k + 4.k + 4.C + C) \cdot \sqrt[k]{L(n)} \cdot T(n)$$

bzw.

$$(16.r.k + 4.k + 4.C + C) \cdot T(n)^{1 + \frac{1}{k}}$$

und Bandbeschränkung $L(n)$ arbeitet.

Vergleicht man im Sinne dessen, was zu Beginn dieses Abschnittes gesagt wurde, die Ergebnisse mit dem entspre-

chenden Satz für Turingmaschinen (jede mehrbändige TM mit Zeitbeschränkung $T(n)$ läßt sich durch eine zweibändige TM mit Zeitbeschränkung $T(n) \cdot \log T(n)$ simulieren), so läßt sich einerseits feststellen, daß der zusätzliche Zeitaufwand, der nötig ist, um bei der Simulation eine größere Anzahl von Marken zu ersetzen, im Verhältnis größer ist als der entsprechende zusätzliche Zeitaufwand bei der TM, der dort nötig ist, um eine größere Anzahl von Bändern zu ersetzen. Andererseits erlaubt Satz 4.4 ein genaueres Anpassen der Zeitbeschränkung an die Markenzahl der simulierenden TAS, ein entsprechender Satz für mehrbändige Turingmaschinen - etwa über die Simulation einer k' -Band-TM durch eine k -Band-TM ($k < k'$) - ist, außer für den Fall $k = 2$, nicht bekannt.

5. Hierarchiesätze für Komplexitätsklassen der TAS

Eine der wichtigsten Fragestellungen bei der Definition eines automatenorientierten Komplexitätsmodells ist die Frage nach einer Hierarchie von Komplexitätsklassen: Um wieviel größer muß die Zeitbeschränkung bzw. Bandbeschränkung eines Automaten A im Vergleich zu einem Automaten B sein, damit es eine Sprache gibt, die zwar von A, nicht aber von B akzeptiert werden kann? Auch die Anzahl der Bänder oder - im Fall der TAS - die Anzahl der Marken kann ein maßgebliches Kriterium für die Existenz einer Hierarchie von Komplexitätsklassen sein. Man versucht dabei, möglichst feine Hierarchien nachzuweisen, etwa daß eine nur wenig größere Zeitbeschränkung des Automaten A im Vergleich zu B den Automaten A schon "mächtiger" macht als den Automaten B.

Zur genaueren Formulierung solcher Zusammenhänge für die TAS soll zunächst der Begriff der Komplexitätsklasse definiert werden.

Definition: Sei Σ ein Alphabet mit mindestens zwei Sym-

bohlen, o.B.d.A. gelte $0,1 \in \Sigma$. Dann werden Komplexitätsklassen $C(\dots)$ wie folgt definiert:

$C_{TAS}^{r,k}(L(n), T(n)) := \{ L \in \Sigma^* \mid \text{Es gibt eine } r\text{-Band-TAS } \hat{T} \text{ mit } k \text{ Marken und eine Konstante } C, \text{ so daß } \hat{T} \text{ mit Bandbeschränkung } C.L(n), \text{ Zeitbeschränkung } C.T(n) \text{ arbeitet und } L \text{ akzeptiert} \}$

$C_{TAS}^{r,k}(T(n)) := \{ L \in \Sigma^* \mid \text{Es gibt eine } r\text{-Band-TAS } \tilde{T} \text{ mit } k \text{ Marken und eine Konstante } C, \text{ so daß } \tilde{T} \text{ mit Zeit-}$

beschränkung C.T(n) arbeitet und L akzeptiert }

Analog werden die Klassen $C_{1TAS}^{r,k}(T(n))$ für die 1TAS mit r Bändern und k Marken und $C_{TM}^{r,k}(T(n))$ für die TM mit r Bändern definiert. Mit $C_{\dots}^{\infty,\infty}$ werden jeweils die entsprechenden Klassen für beliebig, aber endlich viele Bänder und Marken bezeichnet, es wird definiert:

$$C_{TAS}^{\infty,\infty}(L(n), T(n)) := \bigcup_{r,k \geq 1} C_{TAS}^{r,k}(L(n), T(n)),$$

$$C_{TAS}^{\infty,\infty}(T(n)) := \bigcup_{r,k \geq 1} C_{TAS}^{r,k}(T(n)).$$

Die einzige bisher bekannte allgemeine Methode, um die Existenz einer Hierarchie von Komplexitätsklassen zu zeigen, ist die Diagonalisierung; nur für Sonderfälle, wie z.B. die 1-Band-TM, sind auch andere Methoden bekannt. Dieses Prinzip liegt auch dem Beweis des folgenden Satzes zugrunde; dabei stellt Korollar 4.5 die dafür nötige Existenz einer "universellen" TAS sicher, die in der Lage ist, jede andere TAS (die beliebig viele Marken haben darf) zu simulieren.

Satz 5.1: Sei k eine natürliche Zahl, $k \geq 2$, seien T_1, T_2 ,

L_1, L_2 Zeit- bzw. Bandfunktionen, das Funktionspaar

$(L_1(n), T_2(n))$ sei TAS-gutartig. Für T_1, T_2, L_1, L_2 gelte

$$\lim_{n \rightarrow \infty} \frac{\sqrt[k]{L_1(n) \cdot T_1(n)}}{T_2(n)} = 0$$

sowie

$$\lim_{n \rightarrow \infty} \frac{L_1(n)}{L_2(n)} = 0.$$

Dann gilt

$$C_{TAS}^{\infty,\infty}(L_1(n), T_1(n)) \subsetneq C_{TAS}^{1,k}(L_2(n), T_2(n)).$$

Beweis: Der Beweis benutzt, ebenso wie die entsprechenden Beweise für Turingmaschinen, die Tatsache, daß sich eine TAS, deren Überföhrungsfunktion sich durch eine endliche Maschinentafel beschreiben läßt, gödelisieren läßt und daß damit eine surjektive Abbildung von $\mathbb{N}$ in die Menge aller TAS's möglich ist. Eine Kette w aus Σ^* (das zugrundeliegende Eingabealphabet enthält nach Voraussetzung die Ziffern 0 und 1) kann damit in der üblichen Weise als Gödelnummer einer TAS aufgefaßt werden, wobei aus der Gödelnummer auch hervorgeht, mit welchem Bandalphabet diese Maschine arbeitet und wieviele Bänder und Marken sie besitzt. Es kann daher eine TAS $\tilde{T}$ konstruiert werden, die folgendermaßen arbeitet:

1) $\tilde{T}$ berechnet zu einem gegebenen Eingabewort w der Länge n $T_2(n)$ sowie $L_1(n)$ auf je einer eigenen Spur des Arbeitsbandes. Dazu sind nach Voraussetzung $C_1 \cdot T_2(n)$ Schritte und $C_2 \cdot L_1(n) \leq C_2 \cdot L_2(n)$ Bandfelder ausreichend.

2) $\tilde{T}$ interpretiert w als Gödelnummer $i = i(w)$ einer TAS $\tilde{T}_i$. $\tilde{T}_i$ kann zwar eine beliebige endliche Anzahl von Marken haben, aber nach Korollar 4.5 gibt es zu $\tilde{T}_i$ eine 1-Band-TAS $\tilde{T}_i^!$, die zu $\tilde{T}_i$ äquivalent ist (vorausgesetzt, ein Bereich von $L_1(n)$ Bandfeldern wird nicht verlassen) und nicht mehr als k Marken besitzt. Die Maschinentafel von $\tilde{T}_i^!$ kann von $\tilde{T}$ effektiv berechnet und, geeignet codiert, auf dem Band gespeichert werden. Wenn $\tilde{T}$ dazu mehr als $T_2(n)$ Schritte oder mehr als $L_1(n)$ Bandfelder benötigt, stoppt $\tilde{T}$ und akzeptiert w nicht.

3) $\tilde{T}$ simuliert das Verhalten von $\tilde{T}_i^!$ bei Eingabe von w . Werden dabei mehr als $L_2(n)$ Bandfelder oder mehr als $T_2(n)$

Schritte von $\mathcal{T}_i^!$ benötigt, so stoppt $\mathcal{T}$ und akzeptiert w nicht, andernfalls akzeptiert $\mathcal{T}$ w genau dann, wenn $\mathcal{T}_i^!$ (oder die dazu äquivalente TAS $\mathcal{T}_i$) w nicht akzeptiert (Diagonalisierung).

$\mathcal{T}$ arbeitet nach 1) - 3) mit Zeitbeschränkung $C_3 \cdot T_2(n)$ und mit Bandbeschränkung $C_2 \cdot L_2(n)$. Außerdem läßt sich der Zeit- und Bandbedarf für die eigentliche Simulation von $\mathcal{T}_i^!$ durch $\mathcal{T}$ folgendermaßen in Abhängigkeit von der Zeitbeschränkung $T_1^!(n)$ und der Bandbeschränkung $L_1^!(n)$ von $\mathcal{T}_i^!$ abschätzen: $\mathcal{T}$ kann auf $C_4(i) \cdot L_1(n)$ Feldern eine TAS simulieren, die mit Bandbeschränkung $L_1(n)$ arbeitet. Dabei ist der Koeffizient C_4 zwar von der TAS $\mathcal{T}_i^!$, d.h. von der Gödelnummer i abhängig, nicht aber von der Länge n der Eingabe für $\mathcal{T}_i^!$, entsprechendes gilt für die noch folgenden Größen $C_{..}(i)$. $\mathcal{T}$ hat k Marken und kann damit auf seinem Band dieselbe hierarchische Markenorganisation aus $k-1$ Marken aufbauen, die $\mathcal{T}_i^!$ benutzt, um auf $L_1(n)$ Bandfeldern zu arbeiten. Die Marke m_k tritt nach Satz 4.4 höchstens fünfmal auf dem Band auf, sie kann also außerdem noch dazu benutzt werden, um die Stelle auf dem Band zu kennzeichnen, auf der die Codierung von $\mathcal{T}_i^!$ gespeichert ist. Ein "Nachsehen" in dieser Codierung zur Berechnung des nächsten Schrittes benötigt also höchstens $C_5(i)$ Schritte. Ebenso ist die Anzahl von Schritten, die zur Berechnung der Codierung von $\mathcal{T}_i^!$ benötigt wird, nur von i abhängig, sie sei $C_6(i)$. Zum Mitzählen der Schritte von $\mathcal{T}_i^!$ und Vergleichen mit $T_2(n)$ benötigt $\mathcal{T}$ höchstens $C_7 \cdot T_2(n)$ Schritte. Die verschiedenen Koeffizienten lassen sich zu einem nur von i abhängigen $C_8(i)$ zusammenfassen, so daß gilt: Ist $T_1^!(n)$

die Zeitbeschränkung und $L_1^!(n)$ die Bandbeschränkung von $\mathcal{T}_1^!$, so arbeitet $\mathcal{T}$ zur Simulation von $\mathcal{T}_1^!$ mit Zeitbeschränkung $\min(C_7 \cdot T_2(n), C_8(i) \cdot T_1^!(n))$ und mit Bandbeschränkung $\min(L_2(n), C_8(i) \cdot L_1^!(n))$.

Nun werde angenommen, es gebe eine TAS $\mathcal{T}'$, die zu $\mathcal{T}$ äquivalent ist und die mit Bandbeschränkung $C' \cdot L_1(n)$ und mit Zeitbeschränkung $C' \cdot T_1(n)$ arbeitet. Wenn i ihre Gödelnummer ist, so wird $w := 0 \dots 0b(i)$ als Eingabewort für $\mathcal{T}'$ gewählt, wobei $|w| = n$, $b(i)$ die binäre Darstellung von i sei. Um 1) und 2) wie angegeben auszuführen, benötigt $\mathcal{T}$ zur Codierung von $\mathcal{T}_1^!$ nur $C_6(i)$ Schritte und zur Simulation von $\mathcal{T}_1^!$, wenn w Eingabewort von $\mathcal{T}$ ist, nach dem oben gesagten und Korollar 4.5 nur $C_8(i) \cdot C' \cdot r \cdot k \cdot \sqrt[k]{L_1(n)} \cdot T_1(n)$ Schritte mit einer weiteren Konstanten C' . Für genügend große n gilt aber (n kann ohne Veränderung von i durch Anfügen von Nullen beliebig vergrößert werden)

$$C_6(i) + C_8(i) \cdot C' \cdot r \cdot k \cdot \sqrt[k]{L_1(n)} \cdot T_1(n) \leq T_2(n)$$

sowie

$$C_8(i) \cdot L_1(n) \leq L_2(n),$$

und es folgt der Widerspruch, daß w von $\mathcal{T}$ genau dann akzeptiert wird, wenn es von $\mathcal{T}'$ nicht akzeptiert wird. Es kann also keine TAS $\mathcal{T}'$ mit dieser Band- und Zeitbeschränkung geben, die zu $\mathcal{T}$ äquivalent ist, Satz 5.1 ist damit bewiesen.

Wird Korollar 4.5 in der schwächeren Form (ohne explizite Einbeziehung der Bandbeschränkung $L(n)$) herangezogen, so erhält man den folgenden Satz:

Satz 5.2: Sei k eine natürliche Zahl, $k \geq 2$, seien T_1, T_2 Zeitfunktionen, das Funktionspaar $(T_1(n), T_2(n))$ sei TAS-gutartig. Für T_1, T_2 gelte

$$\lim_{n \rightarrow \infty} \frac{T_1(n)^{1 + \frac{1}{k}}}{T_2(n)} = 0.$$

Dann gilt

$$C_{TAS}^{\infty, \infty}(T_1(n)) \subsetneq C_{TAS}^{1, k}(T_2(n)).$$

Der Beweis verläuft analog zum Beweis von Satz 5.1, an Stelle von L_1, L_2 wird jeweils T_1 und T_2 verwendet, wegen $L_i(n) \leq T_i(n)$ ($i=1,2$) ist das möglich.

Durch eine leichte Abschwächung von Satz 5.2 lassen sich die Zusammenhänge zwischen den Komplexitätsklassen auch in der folgenden Weise beschreiben:

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{T_2(n)} = 0 \quad \Rightarrow \quad C_{TAS}^{\infty, \infty}(T_1(n)) \subsetneq C_{TAS}^{1, k}(T_2(n)^{1 + \frac{1}{k}})$$

Denn setzt man in Satz 5.2 $T'_2(n) := T_2(n)^{1 + \frac{1}{k}}$, also

$$T_2(n) = \frac{T'_2(n)}{k \sqrt[k]{T_2(n)}}, \text{ so gilt}$$

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{T_2(n)} = \lim_{n \rightarrow \infty} \frac{T_1(n) \cdot k \sqrt[k]{T_2(n)}}{T'_2(n)} > \lim_{n \rightarrow \infty} \frac{T_1(n)^{1 + \frac{1}{k}}}{T'_2(n)},$$

woraus nach Satz 5.2

$$C_{TAS}^{\infty, \infty}(T_1(n)) \subsetneq C_{TAS}^{1, k}(T'_2(n)) = C_{TAS}^{1, k}(T_2(n)^{1 + \frac{1}{k}})$$

folgt.

Diese Form soll im folgenden benutzt werden, um die Aussage von Satz 5.2 für feste Markenzahlen durch die Anwendung von Transformationen noch zu verschärfen. Diese Methode wurde von verschiedenen Autoren benutzt und von P.Seife-

ras [Sei 74] explizit formuliert; von ihm sind - für die TAS abgewandelt - die beiden folgenden Lemmata übernommen.

Definition: Sei L eine Sprache über einem Alphabet Σ , $1 \in \Sigma$, $\# \in \Sigma$. Dann wird für eine Funktion $p: \mathbb{N} \rightarrow \mathbb{N}$ mit $p(n) > n$ definiert: $p(L) := \{ w \mid w = v \# 1 \dots 1, v \in L, |w| = p(|v|) \}$. Diese Transformation von L in $p(L)$ wird oft auch als "Padding" bezeichnet.

Lemma 5.3 ([Sei 74], Lemma 10): Sei $p: \mathbb{N} \rightarrow \mathbb{N}$ eine TAS-gutartige Funktion mit $p(n) > n$. Dann gilt für alle $k \geq 2$, $r \geq 1$, für jede Sprache L und jede Zeitbeschränkung $T(n) \geq n$:

$$p(L) \in C_{TAS}^{r,k}(T(n)) \iff L \in C_{TAS}^{r,k}(T(p(n)))$$

Beweis: $\Rightarrow$) Es gebe nach Voraussetzung eine TAS $\mathcal{T}$ (r Bänder, k Marken, die $p(L)$ mit Zeitbeschränkung $T(n)$ akzeptiert. Dann wird eine TAS $\mathcal{T}'$ so konstruiert, daß sie für ein Eingabewort v der Länge n zunächst $w = v \# 1 \dots 1$ mit $|w| = p(|v|)$ auf ein Band schreibt. Nach Voraussetzung kann das in $C_1 \cdot p(|v|)$ Schritten geschehen. Anschließend simuliert $\mathcal{T}'$ die TAS $\mathcal{T}$ mit der Eingabe w ; die Simulation ist in Realzeit möglich, da beide über die gleiche Anzahl von Bändern und Marken verfügen. Insgesamt akzeptiert $\mathcal{T}'$ damit L mit Zeitbeschränkung $C_1 \cdot p(n) + T(p(n)) \leq C_2 \cdot T(p(n))$.

$\Leftarrow$) Es gebe eine TAS $\mathcal{T}$, die L mit Zeitbeschränkung $T(p(n))$ akzeptiert. Dann wird $\mathcal{T}'$ so konstruiert, daß $\mathcal{T}'$ für ein Eingabewort w zunächst überprüft, ob w die Form $v \# 1 \dots 1$ mit $|w| = p(|v|)$ hat; nach Voraussetzung ist das in $C_3 \cdot |w|$ Schritten möglich, da in diesem Fall das Zeichen $\#$ in dem Wort w nur einmal vorkommen darf und $p(|v|)$ aus v in

$C_3 \cdot p(|v|)$ Schritten berechenbar ist. Anschließend simuliert $\mathcal{T}'$ die TAS $\mathcal{T}$ für die Eingabe v , dafür sind höchstens $T(p(|v|))$ Schritte nötig. Insgesamt akzeptiert $\mathcal{T}'$ damit $p(L)$ mit Zeitbeschränkung $C_4 \cdot T(|w|)$.

Lemma 5.4 ([Sei 74], Lemma 11): Seien T_1, T_2 sowie $p_0(n) > n, \dots, p_s(n) > n$ Zeitfunktionen, die in Linearzeit von einer 1-Band-TAS mit k Marken berechenbar sind ($k \geq 2$), es gelte $T_1(p_{i+1}(n)) \leq T_2(p_i(n))$ für alle $0 \leq i < s$; weiter gelte $r \geq 1$ und $T_2(n) \geq T_1(n)$ für fast alle n . Dann folgt aus

$$C_{TAS}^{r,k}(T_1(p_0(n))) \not\subseteq C_{TAS}^{r,k}(T_2(p_s(n)))$$

auch

$$C_{TAS}^{r,k}(T_1(n)) \not\subseteq C_{TAS}^{r,k}(T_2(n)) .$$

Beweis: Wegen $T_1(p_{i+1}(n)) \leq T_2(p_i(n))$ gilt

$$C_{TAS}^{r,k}(T_1(p_{i+1}(n))) \subseteq C_{TAS}^{r,k}(T_2(p_i(n))) .$$

Würde nun für alle i ($0 \leq i < s$) gelten

$$C_{TAS}^{r,k}(T_2(p_i(n))) \subseteq C_{TAS}^{r,k}(T_1(p_i(n))) ,$$

so würde daraus folgen

$$\begin{aligned} C_{TAS}^{r,k}(T_2(p_s(n))) &\subseteq C_{TAS}^{r,k}(T_1(p_{s-1}(n))) \\ &\subseteq C_{TAS}^{r,k}(T_2(p_{s-2}(n))) \\ &\subseteq C_{TAS}^{r,k}(T_1(p_{s-2}(n))) \\ &\vdots \\ &\subseteq C_{TAS}^{r,k}(T_1(p_0(n))) , \end{aligned}$$

im Widerspruch zur Voraussetzung. Es muß also für ein i eine Sprache L geben, die in $C_{TAS}^{r,k}(T_2(p_i(n)))$, nicht aber in $C_{TAS}^{r,k}(T_1(p_i(n)))$ enthalten ist. Nach Lemma 5.3 gilt dann aber für $p_i(L)$, daß $p_i(L)$ in $C_{TAS}^{r,k}(T_2(n))$, nicht aber in

$C_{TAS}^{r,k}(T_1(n))$ enthalten ist. Es gilt also dann

$$C_{TAS}^{r,k}(T_1(n)) \not\subset C_{TAS}^{r,k}(T_2(n)) ,$$

was zu beweisen war.

Für den Fall, daß die Zeitbeschränkungen $T_1(n), T_2(n)$ Funktionen sind, die sich als n^α, n^β darstellen lassen, läßt sich daraus folgern:

Satz 5.5: Sei $T_1(n) = n^\alpha$, $T_2(n) = n^\beta$ mit $\alpha, \beta \in \mathbb{R}$, $\alpha, \beta \geq 1$, das Funktionspaar $(T_1(n), T_2(n))$ sei TAS-gutartig. Es gelte für ein reelles $\varepsilon > 0$

$$\lim_{n \rightarrow \infty} \frac{T_1(n)^{1+\varepsilon}}{T_2(n)} = 0 ,$$

d.h. es sei $\beta > \alpha(1+\varepsilon)$. Dann gilt

$$C_{TAS}^{r,k}(T_1(n)) \not\subset C_{TAS}^{r,k}(T_2(n)) \quad \text{für alle } r \geq 1, k \geq 2.$$

Beweis: Es wird eine natürliche Zahl s so gewählt, daß $\frac{\alpha}{2 \cdot s} < \varepsilon$ ist. Dann wird Lemma 5.4 angewandt, wobei

$$p_i(n) := n^{1 + \frac{1}{2 \cdot s}}$$

gesetzt wird. Es gilt dann (im vorliegenden Fall ist

$$T_j(p_i(n)) = p_i(T_j(n)) \quad \text{für } j=1,2)$$

$$T_1(p_{i+1}(n)) = n^{\alpha \cdot (1 + \frac{1}{2 \cdot s})} = n^{\alpha + \frac{\alpha \cdot 1}{2 \cdot s} + \frac{\alpha}{2 \cdot s}} ,$$

$$\begin{aligned} T_2(p_i(n)) &= n^{\beta \cdot (1 + \frac{1}{2 \cdot s})} > n^{(\alpha + \frac{\alpha}{2 \cdot s})(1 + \frac{1}{2 \cdot s})} \\ &= n^{\alpha + \frac{\alpha \cdot 1}{2 \cdot s} + \frac{\alpha}{2 \cdot s} + \frac{\alpha \cdot 1}{4 \cdot s^2}} , \end{aligned}$$

also $T_1(p_{i+1}(n)) < T_2(p_i(n))$. Zu gegebenem n ist $n^{\frac{1}{2 \cdot s}}$

nach Lemma 4.3 in $C_1 \cdot n^{1 + \frac{1}{2 \cdot s}}$ Schritten berechenbar,

$$n^{\frac{i}{2 \cdot s}} = n^{\frac{1}{2 \cdot s}} \cdot n^{\frac{1}{2 \cdot s}} \dots n^{\frac{1}{2 \cdot s}} \quad (i\text{-mal}) \quad \text{dann nach Lemma 4.2 in}$$

$C_2 \cdot (n^{\frac{1}{2 \cdot s}})^i = C_2 \cdot n^{\frac{i}{2 \cdot s}}$ Schritten. Damit ist $p_i(n) = n^{1 + \frac{i}{2 \cdot s}} =$
 $= n \cdot n^{\frac{i}{2 \cdot s}}$ in $C_3 \cdot n^{1 + \frac{i}{2 \cdot s}}$ Schritten berechenbar, also TAS-
gutartig. Außerdem gilt

$$T_1(p_0(n)) = T_1(n),$$

$$T_2(p_s(n)) = T_2(n^{1 + \frac{1}{2}}) = (T_2(n))^{1 + \frac{1}{2}},$$

also nach Satz 5.2

$$C_{TAS}^{r,k}(T_1(p_0(n))) \not\subseteq C_{TAS}^{r,k}(T_2(p_s(n))) .$$

Nach Lemma 5.4 folgt daraus

$$C_{TAS}^{r,k}(T_1(n)) \not\subseteq C_{TAS}^{r,k}(T_2(n)) .$$

Die bisherigen Sätze zeigten die Existenz einer Hierarchie von Komplexitätsklassen in Abhängigkeit von einer Vergrößerung der Zeitbeschränkung, wobei im allgemeinen die "mächtigere" Komplexitätsklasse sogar mit einer geringeren Anzahl von Marken auskommt. In Ergänzung dazu soll nun gezeigt werden, daß unter sehr schwachen Bedingungen für die Zeitbeschränkung die Existenz einer zusätzlichen Marke zu neuen Komplexitätsklassen führt.

Satz 5.6: Seien T_1, T_2 Zeitfunktionen mit

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{T_2(n)} = 0,$$

$T_2(n)$ sei TAS-gutartig. Dann gilt:

- (1) $C_{TAS}^{r,k}(T_1(n)) \not\subseteq C_{TAS}^{r,k+1}(T_2(n))$ für jedes $r \geq 1, k \geq 1$
- (2) $C_{TM}^r(T_1(n)) \not\subseteq C_{TAS}^{1,1}(T_2(n))$ für jedes $r \geq 1$
- (3) $C_{TM}^r(T_1(n)) \not\subseteq C_{TAS}^{1,2}(T_2(n))$ "

Beweis: Der Beweis wird ebenso wie bei Satz 5.1 durch Diagonalisierung geführt und soll deshalb in den Einzelheiten hier nicht wiederholt werden. Voraussetzung für eine Diagonalisierung ist jeweils, daß es eine universelle Maschine der "mächtigeren" Klasse gibt, die in der Lage ist, jede Maschine der "schwächeren" Klasse zu simulieren.

Für (1) ist das unmittelbar einsichtig, denn eine TAS $\tilde{T}_2$ mit $k+1$ Marken kann mit k Marken das Verhalten einer TAS $\tilde{T}_1$, die k Marken besitzt, simulieren. Die zusätzliche Marke wird dazu verwendet, um in einem Schritt an die Codierung von $\tilde{T}_1$ heranzukommen, die sich $\tilde{T}_2$ zu Beginn der Rechnung auf einer Spur des Bandes notiert hat. Das Nachsehen in der Codierung von $\tilde{T}_1$ erfordert also eine Anzahl von Schritten, die nur von der Gödelnummer von $\tilde{T}_1$ abhängt, und ebenso wie bei Satz 5.1 läßt sich zeigen, daß die Bedingung $\lim_{n \rightarrow \infty} \frac{T_1(n)}{T_2(n)} = 0$ ausreicht, um die Simulierbarkeit jeder TAS $\tilde{T}_1$ mit beliebiger Gödelnummer und Zeitbeschränkung $T_1(n)$ durch eine universelle TAS $\tilde{T}_2$ mit Zeitbeschränkung $T_2(n)$ zu gewährleisten.

Der Beweis von (2) und (3) verläuft analog: Die Codierung einer TM $\tilde{T}_1$ wird durch eine Marke in einem Schritt zugreifbar gehalten, für das Hin- und Herspringen von $\tilde{T}_2$ zwischen Codierung von $\tilde{T}_1$ und bearbeitetem Bandabschnitt genügt bei der TAS eine Marke, während eine 1TAS dazu zwei Marken benötigt, die, wie in Abschnitt 2 (S. 15) skizziert, im Wechsel dazu dienen, um auf Codierung oder bearbeiteten Bandabschnitt in Sprüngen vom Bandanfang her zugreifen zu können.

Für die 1TAS läßt sich die Aussage (1) von Satz 5.6 sogar noch verschärfen:

Satz 5.7: Seien T_1, T_2 Zeitfunktionen mit

$$\lim_{n \rightarrow \infty} \frac{T_1(n)}{T_2(n)} = 0 ,$$

$T_2(n)$ sei TAS-gutartig. Dann gilt

$$C_{1TAS}^{r,k}(T_1(n)) \subsetneq C_{1TAS}^{r,k}(T_2(n)) \quad \text{für jedes } r \geq 1, k \geq 2.$$

Beweis: Der Beweis verläuft ebenso wie bei Satz 5.6 durch Diagonalisierung; bei der 1TAS ist jedoch, wenn eine 1TAS $\mathcal{T}_1$ durch eine 1TAS $\mathcal{T}_2$ zu simulieren ist, keine zusätzliche Marke nötig, damit $\mathcal{T}_2$ in einem Schritt auf die Codierung von $\mathcal{T}_1$ zugreifen kann. Denn hier kann die Codierung der 1TAS $\mathcal{T}_1$, die zu simulieren ist, immer am Bandanfang gespeichert und mit einer Marke m_1 markiert werden, der Zugriff auf die Codierung ist dann in einem Schritt der simulierenden 1TAS $\mathcal{T}_2$ möglich. Soll die Marke m_1 an anderer Stelle im Verlauf der Simulation als Sprungziel dienen, so wird zunächst die Marke m_1 am Bandanfang gelöscht und statt dessen eine Marke m_2 gesetzt, der nächste Sprung erfolgt dann auf das nächste, "eigentliche" Exemplar der Marke m_1 . Für die nächsten Simulationsschritte ist der Bandanfang nun mit m_2 markiert, bis m_2 wieder einmal durch m_1 ersetzt wird usw.

6. Beziehungen zwischen TAS und RAM

Wenn die TAS den Anspruch erfüllen soll, ein realistisches Modell für die Berechnung des Aufwands eines Algorithmus darzustellen - zumindest ein realistischeres als die normale Turingmaschine -, so sollten die Komplexitätsklassen der TAS nicht zu weit entfernt von denen der Random-Access-Maschine (RAM) liegen, die allgemein als das abstrakte Automatenmodell angesehen wird, das realen Computern relativ am nächsten kommt. Die folgenden Sätze zeigen, daß das für die TAS zutrifft. Dabei wird (etwas modifiziert) das Modell der RAM zugrundegelegt, das von B.Monien in [Mo 74 a,b] eingeführt wurde, es entspricht bis auf die Additionsoption dem Modell von S.A.Cook [CR 73].

Das Programm einer Random-Access-Maschine (RAM) ist definiert als eine endliche Folge von Befehlen, die auf einer unendlichen Folge von Registern operieren, wobei die folgenden Befehle möglich sind:

- | | |
|-----------------------------|--|
| 1) $X_i \leftarrow X_j$ | Transportbefehl |
| 2) $X_i \leftarrow R_{X_j}$ | } Transportbefehle mit indirekter
Adressierung |
| 3) $R_{X_i} \leftarrow X_j$ | |
| 4) $X_i \leftarrow X_i + 1$ | } arithmetische Operationen |
| 5) $X_i \leftarrow X_i - 1$ | |
| 6) TRA m if $X_j > 0$ | bedingter Sprung: Wenn $X_j > 0$ ist,
wird als nächstes der m-te Befehl
des Programms ausgeführt |
| 7) Read X_i | } Ein/Ausgabe |
| 8) Print X_i | |

Die Bedeutung dieser Befehle ist offensichtlich. $X_i \leftarrow R_{X_j}$ bewirkt, daß der Inhalt des Registers, dessen Adresse sich im j -ten Register befindet, in das i -te Register übertragen wird. Im Hinblick auf eine spätere Verallgemeinerung wird in dieser Definition unterschieden zwischen Registern, deren Adressen im Programm explizit vorkommen (X_i), und den nur indirekt adressierbaren Registern (R_i); wegen der endlichen Länge jedes Programms gibt es jeweils nur endlich viele Register X_i ¹⁾. Jedes Register kann eine beliebig große Zahl speichern; dadurch, daß aber als arithmetische Operation nur die Addition bzw. Subtraktion einer 1 möglich ist, können in t Schritten Zahlen höchstens bis zur Größe $t + C$ (C Konstante, die von den Eingabedaten abhängt) berechnet werden. Hierin liegt der wesentliche Unterschied zu dem sonst äquivalenten Modell von Cook.

Bei der Berechnung einer RAM sind zu Beginn alle Register mit 0 besetzt, eine Folge von Zahlen $k_1, \dots, k_n, 0$ mit $k_j \in \mathbb{N}$ dient als Eingabe. Operationen der Form $\text{Read } X_i$ bewirken, daß die jeweils nächste der Zahlen k_j eingelesen und in das i -te Register gebracht wird. Eine Funktion $f: \mathbb{N}^n \rightarrow \mathbb{N}$ heißt RAM-berechenbar mit Zeitbeschränkung $T(n)$, wenn es eine RAM $\hat{R}$ gibt, die für alle Eingaben $z_1, \dots, z_n, 0$ mit $z_j \in \mathbb{N}$ nach höchstens $T(n)$ Schritten hält und als Ausgabe $f(z_1, \dots, z_n)$ druckt. Eine Funktion $f: \mathbb{N} \rightarrow \mathbb{N}$ heißt RAM-

1) Dem allgemeinen Sprachgebrauch für die RAM folgend, wird hier nicht immer exakt zwischen dem Register X_i (genauer: dem i -ten direkt adressierbaren Register) und X_i als dem Inhalt dieses Registers unterschieden; aus dem Zusammenhang wird aber jeweils deutlich, was gemeint ist.

gutartig, wenn es eine Konstante C gibt, so daß $f(n)$ mit Zeitbeschränkung $C \cdot f(n)$ RAM-berechenbar ist.

In [Mo 74a] und [Fl 74] wurden zwei Erweiterungen des Modells der RAM untersucht, die RAM mit k Speichern und die RAM mit k-dimensionalem Speicher.

Eine RAM mit k Speichern verfügt über k Folgen von indirekt adressierbaren Registern, die Transportbefehle mit indirekter Adressierung haben hier also die Form

$$\begin{aligned} X_i &\leftarrow R_{X_j}^{(m)} & (1 \leq m \leq k) \\ R_{X_i}^{(m)} &\leftarrow X_j \end{aligned}$$

In [Mo 74a] wird gezeigt:

Lemma 6.1: Zu jeder RAM mit k Speichern und Zeitbeschränkung $T(n)$ gibt es eine äquivalente RAM mit nur einem Speicher und Zeitbeschränkung $C \cdot k \cdot T(n)$.

Eine RAM mit k-dimensionalem Speicher verfügt über eine als k-dimensionale, unendliche Matrix angeordnete Menge von indirekt adressierbaren Registern, die Transportbefehle mit indirekter Adressierung haben hier die Form

$$\begin{aligned} X_i &\leftarrow R(X_{i_1}, \dots, X_{i_k}) \\ R(X_{i_1}, \dots, X_{i_k}) &\leftarrow X_j \end{aligned}$$

In [Fl 74] wird gezeigt:

Lemma 6.2: Zu jeder k-dimensionalen RAM ($k \geq 2$) mit Zeitbeschränkung $T(n)$ gibt es eine äquivalente zweidimensionale RAM mit Zeitbeschränkung $C \cdot T(n)$, zu jeder zweidimensionalen RAM mit Zeitbeschränkung $T(n)$ gibt es eine äquivalente eindimensionale RAM mit Zeitbeschränkung $C \cdot T(n) \cdot \log T(n)$.

Das Akzeptieren einer Sprache über einem Alphabet

$\Sigma = \{a_1, \dots, a_p\}$ wird in Analogie zur TM und zur TAS folgendermaßen definiert: Eine RAM $\tilde{R}$ akzeptiert eine Sprache $L \subseteq \Sigma^*$ genau dann, wenn $\tilde{R}$ für Eingabefolgen $i_1, \dots, i_n, 0$ mit $i_j \leq p$ genau dann hält und als letzte Ausgabe vor dem Halten eine 0 ausgibt, wenn $a_{i_1} \dots a_{i_n} \in L$ gilt. $\tilde{R}$ akzeptiert $L \subseteq \Sigma^*$ mit Zeitbeschränkung $T(n)$, wenn L von $\tilde{R}$ akzeptiert wird und $\tilde{R}$ für jede Eingabe $i_1, \dots, i_n, 0$ mit $1 \leq i_j \leq p$ nach höchstens $T(n)$ Schritten hält. Komplexitätsklassen für die RAM (hier nur im Hinblick auf die Spracherkennung) werden ebenfalls analog zur TM und zur TAS definiert:

$C_{RAM}^r(T(n)) := \{L \subseteq \Sigma^* \mid \text{Es gibt eine } r\text{-dimensionale RAM } \tilde{R} \text{ und eine Konstante } C, \text{ so daß } \tilde{R} \text{ mit Zeitbeschränkung } C \cdot T(n) \text{ arbeitet und } L \text{ akzeptiert}\}$

Der folgende Satz stammt von B. Monien ([WM 73], Teil 2), im Beweis wird dieselbe Idee wie bei Satz 4.1 und Satz 4.4 benutzt.

Satz 6.3: Sei $L \subseteq \Sigma^*$ eine Sprache, die von einer RAM $\tilde{R}$ mit Zeitbeschränkung $T(n)$ akzeptiert wird, $T(n)$ sei TAS-gutartig. Dann gibt es zu jeder natürlichen Zahl k , $k \geq 2$, eine zu $\tilde{R}$ äquivalente TAS $\tilde{T}_k$ mit $k+1$ Marken, die L mit Zeitbeschränkung $C \cdot k \cdot T(n)^{1 + \frac{1}{k}}$ akzeptiert.

Beweis: Die Grundidee des Beweises ist, daß sich $\tilde{T}_k$ durch hierarchische Anordnung der k Marken eine schnelle Zugriffsmöglichkeit auf Gruppen von Bandfeldern schafft, in denen jeweils der Inhalt eines Registers von $\tilde{R}$ gespeichert ist.

Dazu geht $\mathcal{T}_k$ wie im folgenden (1) - 5)) beschrieben vor:

1) $\mathcal{T}_k$ berechnet bei gegebener Eingabe $a_{i_1} \dots a_{i_n}$ zunächst $T(n)$ und schreibt es unär auf eine Spur des Bandes. Nach Voraussetzung kommt $\mathcal{T}_k$ dazu mit 2 Marken und $C_1 \cdot T(n)$ Schritten aus.

2) $\mathcal{T}_k$ berechnet die Zahl $Z_k := \sqrt[k]{T(n)}$. Nach Lemma 4.3 reichen dafür $C_2 \cdot Z_k^{k+1} \leq (\sqrt[k]{T(n)} + 1)^{k+1} \leq C_3 \cdot T(n)^{1 + \frac{1}{k}}$ Schritte aus.

3) $\mathcal{T}_k$ berechnet $L_k := \lceil \log Z_k \rceil$. Dies geschieht durch sukzessives Berechnen von 2, 4, 8, ... und Vergleich mit Z_k . Der Zeitaufwand dafür läßt sich nach Lemma 4.2 durch

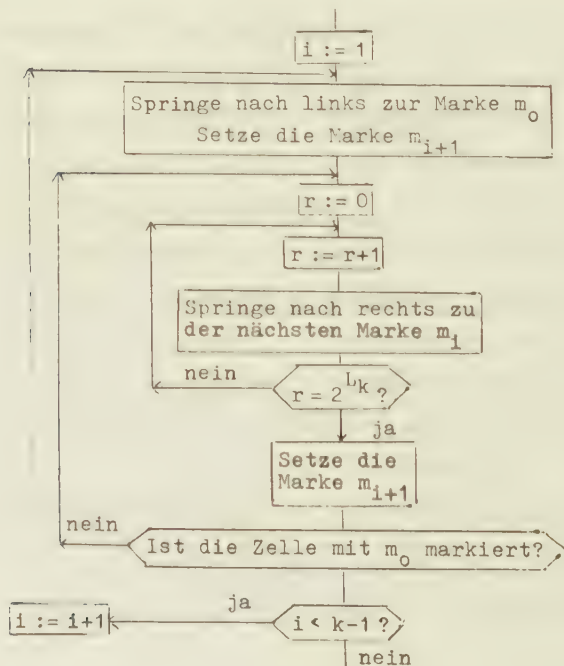
$$C_4 \cdot \sum_{j=1}^{L_k} 2^j \leq C_4 \cdot 2^{L_k+1} \leq C_5 \cdot T(n)^{\frac{1}{k}}$$

abschätzen.

4) Die Zahlen, die im Verlauf der Rechnung von $\hat{R}$ berechnet werden, sind durch $C_6 + T(n)$ beschränkt (o.B.d.A. sei $C_6 = 0$, andernfalls wäre bei der folgenden Rechnung noch eine weitere Konstante mitzuführen). Jede solche Zahl kann von $\mathcal{T}_k$ in Binärschreibweise in $k \cdot L_k$ Bandfeldern gespeichert werden. Solche Bandabschnitte von je $k \cdot L_k$ Feldern werden im folgenden in Analogie zur RAM als "Register" bezeichnet; um einen Zugriff zum jeweils nächsten "Register" in einem Schritt zu ermöglichen, markiert $\mathcal{T}_k$ auf dem Band jedes $(i \cdot k \cdot L_k + 1)$ -te Feld (also das erste Feld jedes "Registers") mit der Marke m_1 . Dies geschieht zunächst r -mal ($i = 0, \dots, r-1$), wenn im Programm von $\hat{R}$ Adressen $1, \dots, r$ explizit vorkommen, und dann noch $2^{k \cdot L_1} - \text{mal}$ ($i = r, \dots, r + 2^{k \cdot L_k}$) für die indirekt adressierbaren Register. Denn Adressen bis zu dieser

und höchstens bis zu dieser Größe können bei einer Berechnung von $\bar{R}$ auftreten. Außerdem wird das erste, das $(r+1)$ -te und das letzte "Register" mit der Marke m_0 markiert. Da die Zahlen k und 2^{L_k} in unärer Darstellung vorliegen, kann die Markierung mit m_1 und m_0 in $C_7 \cdot 2^{k \cdot L_k} \cdot k \cdot L_k \leq C_8 \cdot T(n) \cdot \log T(n)$ Schritten geschehen.

5) Zusätzlich werden, um einen schnelleren Zugriff auf die "Register" zu ermöglichen, noch Marken $m_2, \dots, m_k$ gemäß der folgenden Bedingung auf dem Band verteilt: Das erste Feld des i -ten "Registers" wird genau dann mit der Marke m_j ($2 \leq j \leq k$) markiert, wenn $2^{j \cdot L_k}$ ein Teiler von i ist. Das Setzen dieser Marken kann nach dem durch das folgende Flußdiagramm beschriebenen Algorithmus erfolgen:



Der Zeitaufwand dafür beträgt

$$C_9 \cdot \sum_{j=0}^k \frac{2^{k \cdot L_k}}{2^{j \cdot L_k}} \leq C_9 \cdot 2^{L_k+1} \leq C_{10} \cdot T(n)$$

Schritte. Die Marken sind dann folgendermaßen verteilt:

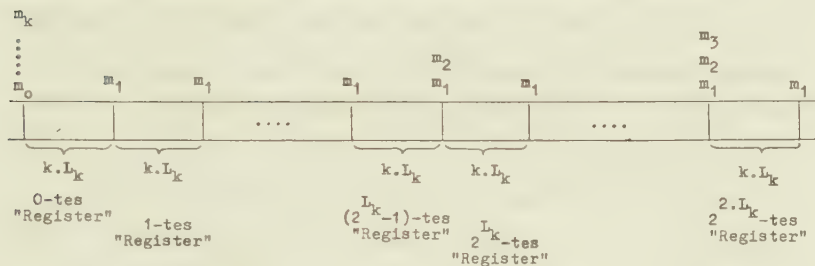


Abb. 15: Markenorganisation von $\mathcal{T}_k$

Zur eigentlichen Simulation der Schritte von $\hat{R}$ durch $\mathcal{T}_k$ sei das Programm von $\hat{R}$ im endlichen Gedächtnis gespeichert. $\mathcal{T}_k$ startet mit $a_{1_1} \dots a_{1_n}$ auf dem Eingabeband und trägt sich während der Simulation in den "Registern" des Arbeitsbandes jeweils den Inhalt der entsprechenden Register von $\hat{R}$ in Binärschreibweise ein, der Zahl 0 als Registerinhalt entspricht dabei ein "Register" von $\mathcal{T}_k$, in dem nur Leerzeichen stehen.

Es soll nun gezeigt werden, wie die einzelnen Operationen von $\hat{R}$ durch $\mathcal{T}_k$ simuliert werden; dabei sind offensichtlich die Transportbefehle mit indirekter Adressierung die einzigen Fälle, die eine eingehendere Betrachtung erfordern:

Sei eine Operation $X \leftarrow R_{X_j}$ von $\hat{R}$ zu simulieren. Durch j Sprünge mit Marken m_1 vom Bandanfang aus nach rechts wird das j -te Register X_j aufgesucht, sein Inhalt ist

eine Zeichenkette $\alpha = \alpha_1 \dots \alpha_{k \cdot L_k} \in \{0,1\}^*$. Teilt man diese Kette in Blöcke β_j ($j=1, \dots, k$) der Länge L_k , so ist offensichtlich die Zahl $\beta = \beta_1 + 2^{L_k} \cdot \beta_2 + 2^{2 \cdot L_k} \cdot \beta_3 + \dots + 2^{(k-1) \cdot L_k} \cdot \beta_k$ der Zahlenwert der Zeichenkette α . $\tilde{\gamma}_k$ überträgt nun nacheinander $\beta_k, \dots, \beta_1$ in Unärschreibweise und führt, vom ersten indirekt adressierbaren "Register" R_1 ausgehend, nacheinander β_k Sprünge nach rechts mit Marken m_k , β_{k-1} Sprünge mit Marken $m_{k-1}, \dots, \beta_1$ Sprünge mit Marken m_1 aus und erreicht auf diese Weise das Register mit der Adresse β , die durch den Inhalt von Register X_j gegeben ist. Die jeweilige Kopfstellung im Verlauf dieser Sprünge kann durch ein weiteres Exemplar der Marke m_0 festgehalten werden, die sonst nur zum Markieren von Anfang und Ende der "Register"-Folge benutzt wird. Die Anzahl von Schritten zum Auffinden von R_{X_j} läßt sich durch $C_{11} \cdot \sum_{j=1}^k \beta_j \leq C_{11} \cdot k \cdot 2^{L_k} \leq C_{12} \cdot k \cdot (T(n)^{\frac{1}{k}} + 1)$ abschätzen, das anschließende Übertragen des Inhalts in das Register X_i benötigt nicht mehr als $C_{13} \cdot k \cdot L_k \leq C_{14} \cdot k \cdot \log T(n)$ Schritte.

Analog kann eine Operation $R_{X_i} \leftarrow X_j$ von $\tilde{\gamma}_k$ in $C_{15} \cdot k \cdot T(n)^{\frac{1}{k}}$ Schritten simuliert werden. Für die anderen Operationen gilt: In $C_{16} \cdot k \cdot L_k \leq C_{16} \cdot k \cdot \log T(n)$ Schritten kann $\tilde{\gamma}_k$ den Inhalt eines "Registers" in ein anderes "Register" übertragen, eine 1 addieren oder subtrahieren. Da es sich hier immer um fest adressierbare Register X_i, X_j handelt, ist der Zugriff zu diesen Registern mit Hilfe der Marken m_1 in einer konstanten Anzahl von Operationen möglich. Jede der Operationen

$X_i \leftarrow X_j$ TRA m if $X_j > 0$
 $X_i \leftarrow X_i + 1$ Read X_i (für Eingaben $\leq C$, C konstant)
 $X_i \leftarrow X_i \div 1$

kann von $\mathcal{T}_k$ also in $C_{17} \cdot k \cdot \log T(n)$ Schritten simuliert werden. Damit kommt $\mathcal{T}_k$ für die Vorbereitung des Bandes mit $C' \cdot k \cdot T(n)^{1 + \frac{1}{k}}$ und für die Simulation jeder einzelnen Operation mit $C'' \cdot k \cdot T(n)^{\frac{1}{k}}$ Schritten, insgesamt also mit $C \cdot k \cdot T(n)^{1 + \frac{1}{k}}$ Schritten und $k+1$ Marken aus, Satz 6.3 ist damit bewiesen.

Satz 6.3 setzt ein RAM-Modell voraus, bei dem als einzige arithmetische Operationen solche der Form $X_i \leftarrow X_i + 1$, $X_i \leftarrow X_i \div 1$ vorkommen (In [Mo 74a] wird gezeigt, daß die Komplexitätsklassen sich nicht ändern, wenn auch auf die Subtraktion verzichtet und nur die Addition einer 1 als arithmetische Operation zugelassen wird). Diese Eigenschaft der hier verwendeten RAM wird aber im Beweis nur dazu genutzt, um daraus zu folgern, daß die berechneten Zahlen sich durch $C + T(n)$ beschränken lassen. Kann andererseits eine solche Beschränkung (wobei auch eine Beschränkung durch $(T(n))^C$ noch zugelassen werden kann) vorausgesetzt werden, so läßt sich der Beweisgang auch auf das Cook'sche Modell der RAM mit allgemeiner Addition und Subtraktion übertragen, und es gilt

Korollar 6.4: Sei L eine Sprache, die von einer RAM $\mathcal{R}$ mit allgemeinem Additions- und Subtraktionsbefehl $(X_i \leftarrow X_j + X_k, X_i \leftarrow X_j - X_k)$ mit Zeitbeschränkung $T(n)$ akzeptiert wird, $T(n)$ sei TAS-gutartig. Für $\mathcal{R}$ gelte jedoch,

daß im Verlauf von $T(n)$ Schritten Zahlen höchstens bis zur Größe $T(n)^C$ (C konstant) berechnet werden. Dann gibt es zu jeder natürlichen Zahl k , $k \geq 2$, eine zu $\mathbb{R}$ äquivalente TAS $\mathcal{T}_k$ mit $k+1$ Marken, die mit Zeitbeschränkung $C' \cdot k \cdot T(n)^{1 + \frac{1}{k}}$ arbeitet.

Beweis: Der Beweis von Satz 6.3 gilt entsprechend (mit einer zusätzlichen multiplikativen Konstanten 2^C); für die einzigen neu hinzutretenden arithmetischen Operationen $X_i \leftarrow X_j + X_k$ und $X_i \leftarrow X_j - X_k$ gilt ebenfalls, daß die simulierende TAS $\mathcal{T}_k$ die Inhalte zweier "Register" in $C_1 \cdot k \cdot L_k \leq C_2 \cdot k \cdot \log T(n)$ Schritten addieren bzw. subtrahieren kann.

Nach dem Ergebnis von Satz 6.3 liegt die Frage nahe, ob umgekehrt jede TAS durch eine RAM etwa in Linearzeit simulierbar ist, denn die RAM erscheint gegenüber der TAS als das wesentlich "stärkere" Maschinenmodell (Speicherung von unbegrenzt großen Zahlen in einem Register, direkte Zugriffsmöglichkeit auf jedes Register über Adressen). Wegen der Möglichkeit der TAS, in beliebig verschachtelter Anordnung Marken zu setzen oder zu löschen, kann eine TAS jedoch nicht einfach durch eine RAM mit Hilfe einer "Buchführung" über die gesetzten Marken in Linearzeit simuliert werden. Die Methode von Satz 4.4 läßt sich jedoch auch hier verwenden, denn zumindest eine TAS in Normalform läßt sich in Linearzeit durch eine RAM simulieren; die gewünschte allgemeine Aussage über die Simulation einer TAS durch eine RAM ergibt sich dann durch Kombination mit Satz 4.4.

Satz 6.4: Zu jeder TAS $\mathcal{T}$ in Normalform mit Parametern k, s und f , die mit Zeitbeschränkung $T(n)$ und mit Bandbeschränkung $L(n)$ arbeitet, gibt es eine äquivalente RAM $\mathcal{R}$ mit Zeitbeschränkung $C \cdot \max(s, f, k) \cdot T(n) + C' \cdot k \cdot L(n)$.

Beweis: Der Überföhrungsfunktion (Maschinentafel) von $\mathcal{T}$ entspricht ein geeignetes Programm für $\mathcal{R}$ mit fest adressierbaren Registern X_i für die Codierung der vorkommenden Konstanten, während der Bandinhalt von $\mathcal{T}$ einschließlich der Markenorganisation durch die indirekt adressierbaren Register R_i von $\mathcal{R}$ simuliert wird. O.B.d.A. kann dabei $\mathcal{R}$ als RAM mit mehreren Speichern angenommen werden, denn nach Lemma 6.1 läßt sich eine solche RAM wiederum durch eine RAM mit nur einem Speicher in Linearzeit simulieren. In Abb. 16 (S. 80) wird gezeigt, wie sich Bandfelder von $\mathcal{T}$ und Register von $\mathcal{R}$ entsprechen (zunächst nur unter Berücksichtigung der festen Marken). Dabei werden die Bandfelder von $\mathcal{T}$ als geeignet durchnumeriert angenommen; wenn dabei $\mathcal{T}$ auf dem Feld mit der Nummer $T(n)$ zu arbeiten beginnt, kann vermieden werden, daß negativ numerierte Bandfelder auftreten.

Jedem Feld des Arbeitsbandes von $\mathcal{T}$ ist bei $\mathcal{R}$ ein Register in einem "Arbeitsspeicher", ein Register in einem "Adreßverweisspeicher" und ein "Registerblock" von $2 \cdot k + 2$ Registern in einem "Adressenspeicher" zugeordnet. Ist b_i das i -te Bandfeld von $\mathcal{T}$, so gilt für den Inhalt des i -ten Registers bzw. Registerblocks von $\mathcal{R}$:

a) Arbeitsspeicher: Jedes Symbol aus dem Bandalphabet von $\mathcal{T}$ und jede Marke läßt sich bei $\mathcal{R}$ durch eine geeignete

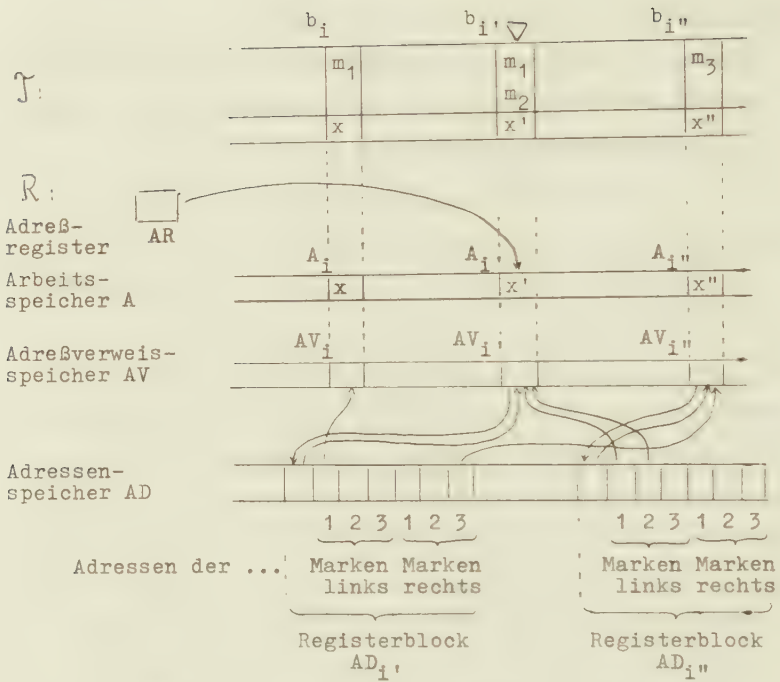


Abb.16: Registerorganisation für feste Marken

Zahl verschlüsseln; im Sinne einer solchen Codierung enthält, wenn bei $\bar{T}$ auf dem i -ten Bandfeld b_i das Symbol x (und evtl. eine Marke) stehen, $\bar{R}$ im i -ten Register A_i des Arbeitsspeichers ebenfalls das Symbol x .

b) Adressenspeicher: Für die einzelnen Register $AD_1^{(0)}, \dots, AD_1^{(2^k-1)}$ des i -ten Registerblocks im Adressenspeicher gilt:

$AD_1^{(0)}$ enthält die Adresse 1 (d.h. die Adresse des zugehörigen Registers im Arbeitsspeicher)

$AD_i^{(j)}$ enthält für $1 \leq j \leq k-1$ die Zahl

$$p = \begin{cases} \max q & q < i, \text{ und } b_q \text{ ist mit Marke } m_j \text{ markiert} \\ 0 & \text{es gibt (noch) kein solches } q \end{cases},$$

für $k \leq k+j-1 \leq 2 \cdot k-2$ ($1 \leq j \leq k-1$) die Zahl

$$p' = \begin{cases} \min q & q > i, \text{ und } b_q \text{ ist mit Marke } m_j \text{ markiert} \\ 0 & \text{es gibt (noch) kein solches } q \end{cases},$$

$AD_i^{(2k-1)}$ enthält eine 1, wenn b_i mit der beweglichen Marke m_k markiert ist, sonst eine 0. Während der Simulation von $\tilde{T}$ durch $\mathcal{R}$ kann $AD_i^{(2k-1)}$ zeitweilig auch anders besetzt werden (s. unten).

c) Adreßverweisspeicher: Das i -te Register AV_i des Adreßverweisspeichers enthält die Adresse des i -ten Registerblocks im Adressenspeicher.

Informell ausgedrückt, enthält also ein Registerblock im Adressenspeicher die Information darüber, wo für jedes j ($1 \leq j \leq k-1$) die nächste (feste) Marke m_j nach links und nach rechts steht; durch den Adreßverweisspeicher wiederum ist für gegebenes i die Information im i -ten Registerblock schnell zugreifbar. Die Adresse i des Registers im Arbeitsspeicher, das dem bei $\tilde{T}$ gerade unter dem Lesekopf stehenden Bandfeld b_i entspricht, wird in einem "Adreßregister" AR gespeichert. Mit Hilfe dieser Registerorganisation sollen in offensichtlicher Weise Sprünge von b_i aus auf die nächste feste Marke simuliert werden können.

Etwas komplizierter ist die Registerorganisation für bewegliche Marken: Zur Simulation von Sprüngen auf bewegliche Marken dient ein weiterer Speicher mit bis zu $f+1$ Blöcken von je drei Registern (f ist der Parameter, der angibt, wie oft bei $\tilde{T}$ die bewegliche Marke m_k auftreten

darf). Jeder solche Block verweist auf eine bewegliche Marke, die Blöcke sind untereinander nach der Technik der verketteten Listen verknüpft; der erste Block ist der jeweils aktuellen Stellung des Lesekopfes zugeordnet. Im einzelnen enthält dabei im "Sprungregister-Speicher" S

$S_1^{(1)}$ die Zahl

- 0, falls das Feld $b_{X_{AR}}$ (das Feld unter dem Lesekopf von $\tilde{J}$) nicht mit m_k markiert ist, bzw.
1, falls $b_{X_{AR}}$ mit m_k markiert ist,

$S_1^{(2)}$ die Adresse des Sprungregisters $S_{j'}^{(1)}$, wenn

$$j' = \begin{cases} \max q & q < X_{AR} \text{ (Inhalt des Adreßregisters), und} \\ & b_q \text{ ist mit Marke } m_k \text{ markiert} \\ 0 & \text{es gibt (noch) kein solches } q \end{cases}$$

$S_1^{(3)}$ die Adresse des Sprungregisters $S_{j''}^{(1)}$, wenn

$$j'' = \begin{cases} \min q & q \geq X_{AR}, \text{ und } b_q \text{ ist mit Marke } m_k \text{ markiert} \\ 0 & \text{es gibt (noch) kein solches } q. \end{cases}$$

Für $1 \leq j \leq f+1$ speichert jeweils

$S_j^{(1)}$ die Adresse i zu einem Bandfeld b_i , wenn b_i mit m_k markiert ist,

$S_j^{(2)}$ die Adresse des Sprungregisters $S_{j'}^{(1)}$, wenn

$$j' = \begin{cases} \max q & q < i = S_j^{(1)}, \text{ und } b_q \text{ ist mit Marke } m_k \\ & \text{markiert} \\ 0 & \text{es gibt (noch) kein solches } q, \end{cases}$$

$S_j^{(3)}$ die Adresse des Sprungregisters $S_{j''}^{(1)}$, wenn

$$j'' = \begin{cases} \min q & q > i = S_j^{(1)}, \text{ und } b_q \text{ ist mit } m_k \text{ markiert} \\ 0 & \text{es gibt (noch) kein solches } q. \end{cases}$$

$S_j^{(2)}$ und $S_j^{(3)}$ verweisen also indirekt auf die Adressen, die den jeweils nächstgelegenen mit m_k markierten Band-

feldern von $\tilde{J}$ entsprechen, wobei "jeweils nächstgelegenen" bedeutet: Für $j = 1$ nächstgelegenen bezüglich der aktuellen Stellung des Lesekopfes von $\tilde{J}$, für $j > 1$ nächstgelegenen bezüglich der Adresse $i = S_j^{(1)}$. Da auf die Blöcke S_j immer nur mit Hilfe der indirekten Adressierung zugegriffen wird, kommt es auf ihre Reihenfolge im Sprungregister-Speicher, wie allgemein bei verketteten Listen, nicht an. Die folgende Abbildung soll diese Registerorganisation noch einmal verdeutlichen:

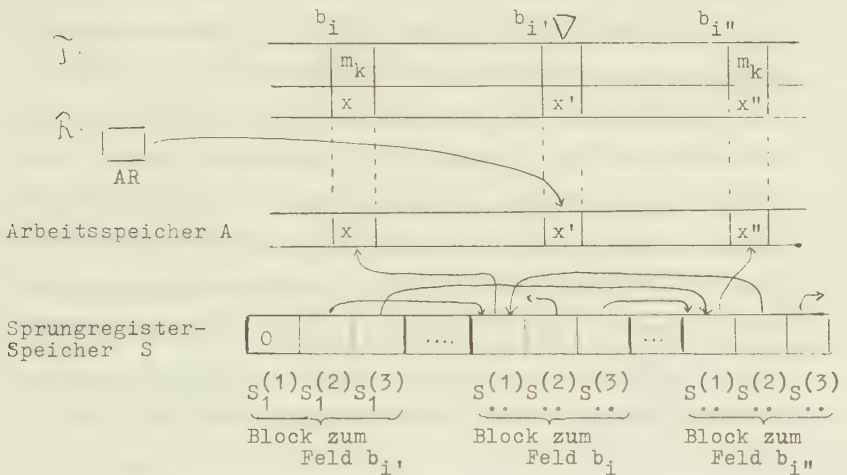


Abb.17: Registerorganisation für bewegliche Marken

Insgesamt werden also die aus der Programmierung bekannten Techniken der indirekten Adressierung (im Adressenspeicher) und der verketteten Listen (im Sprungregister-Speicher) verwendet, um Sprünge von $\tilde{J}$ auf feste bzw. bewegliche Marken zu simulieren.

Zu dieser Simulation von $\tilde{J}$ durch $\hat{R}$ sind nun mehrere Aspekte zu berücksichtigen:

- 1) Die normalen (Turingmaschinen-)Funktionen von $\tilde{J}$,
- 2) das Setzen oder Löschen von beweglichen Marken m_k ,
- 3) Sprünge auf feste oder bewegliche Marken,
- 4) die Initialisierung der Registerorganisation im Adressenspeicher (Setzen von festen Marken).

1) "Normale" Funktionen von $\tilde{J}$: Sei ein Schritt von $\tilde{J}$ zu simulieren, bei dem der Lesekopf auf dem Bandfeld b_i steht. Nach Definition der TAS schließt jeder Schritt, wie bei Turingmaschinen üblich, mit dem Lesen des Inhalts von b_i auch das Prüfen einer Bedingung ein. Da $\tilde{K}$ als Abfrage von Bedingungen nur die Prüfung von Registern auf positiven Inhalt kennt, wird der entsprechende Registerinhalt A_i im Arbeitsspeicher durch sukzessives Subtrahieren einer 1 ermittelt. Der Zustand z von $\tilde{J}$ kann von $\tilde{K}$ dadurch verschlüsselt werden, daß die Kontrolle jeweils bei einem für z charakteristischen Teilprogramm liegt. In Abhängigkeit davon und von A_i wird von $\tilde{K}$ also A_i neu besetzt (entsprechende Konstanten seien in den direkt adressierbaren Registern von $\tilde{K}$ gespeichert), und die Kontrolle geht an ein anderes Teilprogramm, entsprechend dem neuen Zustand von $\tilde{J}$, über. Wenn der zu simulierende Schritt von $\tilde{J}$ kein Sprung ist, dann ergibt sich je nach Bewegung des Lesekopfes von $\tilde{J}$ der neue Inhalt des Adreßregisters AR durch Addition bzw. Subtraktion einer 1. Da Zustandszahl und Bandalphabet von $\tilde{J}$ endlich sind, läßt sich die Schrittzahl von $\tilde{K}$ durch $C_1 \cdot s$ (s = Alphabetgröße von $\tilde{J}$) abschätzen.

- 2) Setzen und Löschen von beweglichen Marken: Muß von $\tilde{J}$

auf dem Feld b_i eine Marke m_k gesetzt oder gelöscht werden, so kann $\bar{R}$ im Register $S_1^{(1)}$ prüfen, ob diese Marke schon gesetzt ist. Wenn nötig, wird dann anschließend in ein neues Sprungregister $S_j^{(1)}$ die Adresse $i = X_{AR}$ eingetragen und die Verkettung der Sprungregister ergänzt (die "nächstgelegenen" Sprungregister sind durch $S_1^{(2)}$ bzw. $S_1^{(3)}$ erreichbar) bzw. durch entsprechende Korrektur in diesen Registern die Marke "gelöscht". Dazu genügt eine konstante Anzahl C_2 von Schritten von $\bar{R}$.

3) Sprünge auf feste oder bewegliche Marken: Simuliert $\bar{R}$ einen Sprung von $\bar{J}$ auf eine bewegliche Marke m_k , so sind dazu die Adressenverweise im ersten Registerblock des Sprungregister-Speichers entsprechend zu verändern; die neu einzutragenden Adressen können mit Hilfe der weiteren Registerblöcke in diesem Speicher bestimmt werden. Dasselbe gilt in dem Fall, daß $\bar{R}$ einen normalen Schritt von $\bar{J}$ simuliert, wenn das neue Bandfeld b_i , mit einer Marke m_k markiert ist (nachprüfbar im Register $AD_{i,1}^{(2k-1)}$). Auch hier ist eine konstante Zahl C_3 von Schritten von $\bar{R}$ ausreichend.

Eine genauere Betrachtung ist jedoch für den Fall nötig, daß $\bar{R}$ einen Sprung von $\bar{J}$ auf eine feste Marke simuliert. Der neue Inhalt X_{AR} des Adreßregisters, der dem neuen Bandfeld von $\bar{J}$ entspricht, kann durch die Registerorganisation des Adressenspeichers in $C_4 \cdot k$ Schritten bestimmt werden. Es ist jedoch noch zu berücksichtigen, daß der Sprung möglicherweise über mehrere bewegliche Marken m_k hinwegführt, daß also die Eintragungen in den Registern $S_1^{(2)}$ und $S_1^{(3)}$, der neuen Stellung des Lesekopfes von $\bar{J}$

4) Schließlich ist noch zu zeigen, wie die beschriebene Registerorganisation initialisiert wird. Die Registerblöcke im Sprungregister-Speicher (nicht mehr als $f+1$) werden im Verlauf der Simulation jeweils neu eingerichtet, so daß hier nur die Eintragung der Adressen im Adressenspeicher für die nächstgelegenen festen Marken zu beschreiben ist: Wenn $\tilde{J}$ eine Marke m_j ($j < k$) auf ein Bandfeld b_i neu setzt, so "läuft" $\tilde{R}$ die benachbarten Felder bis zum nächstgelegenen Feld b_i , mit schon gesetzter Marke m_j "ab" und trägt in die entsprechenden Register $AD_{..}^{(k+j-1)}$ die Adresse i als Sprungziel (hier: Sprungziel nach rechts ein ¹⁾). Danach läuft $\tilde{R}$ wieder zum Feld b_i zurück und trägt in die Register $AD_{..}^{(j)}$ die Adresse i' als Sprungziel bzgl. m_j nach links ein. Dieses Verfahren zum Eintrag der Sprungadressen ist der Grund für die Bedingung (3) (S. 49) bei der Definition der Normalform; sie gewährleistet, daß jedes Feld in dieser Weise beim Setzen von festen Marken höchstens $2(k-1)$ -mal durchlaufen wird. Denn immer werden die "endgültigen" Sprungadressen eingetragen, und jede Sprungadresse wird erst nach ihrem Eintrag für die Simulation eines Sprunges benötigt. Damit sind zur Initialisierung der Registerorganisation höchstens $C_G \cdot k$ zusätzliche Schritte von $\tilde{R}$ pro benutztem Bandfeld von $\tilde{J}$ nötig.

1) Damit $\tilde{R}$ dabei weiß, in welcher Richtung ein entsprechender Lauf von $\tilde{J}$ zu simulieren ist, wird angenommen, daß das Ausgangsfeld b^* von $\tilde{J}$ zu Beginn mit allen Marken $m_1, \dots, m_{k-1}$ markiert wird, daß $\tilde{R}$ sich bei der Simulation merkt, ob rechts oder links von b^* operiert wird, und daß $\tilde{R}$ bei der Eintragung von Sprungzielen für eine neue Marke in die benachbarten Registerblöcke immer die Richtung aufsucht, die bei $\tilde{J}$ der Richtung auf das Ausgangsfeld b^* zu entspricht.

Aus 1) - 4) folgt insgesamt, daß $\bar{T}$ mit Zeitbeschränkung $C \cdot \max(s, f, k) \cdot T(n) + C' \cdot k \cdot l(n)$ auskommt, Satz 6.4 ist damit bewiesen.

Nach diesem etwas langwierigen Beweis soll noch angedeutet werden, wo die Schwierigkeit liegt, wenn man versucht, die Konstruktionsidee des Beweises auf die Simulation einer allgemeinen TAS durch eine RAM zu übertragen. Es wäre zwar, wie das im Beweis geschieht, zunächst möglich, in einem Registerblock der RAM die Adressen der jeweils nächstgelegenen Marken $m_1, \dots, m_k$ zu speichern, man könnte dann bei der Simulation diese Felder (bzw. die entsprechenden Register) in $O(k)$ Schritten erreichen. Wird aber auf einem Feld b_i eine Marke gesetzt oder gelöscht, so wären nicht nur in den zu b_i nächstgelegenen markierten Feldern $b_{i'}$, die entsprechenden Verweise zu ändern, sondern in allen Feldern $b_{i''}$, von denen aus b_i nächstgelegenes markiertes Feld ist. Die Anzahl dieser Felder ist aber im allgemeinen Fall nicht durch eine Konstante beschränkt; wenn b_i mit m_j markiert ist und $b_{i'}$ nächstgelegenes mit $m_{j'}$ markiertes Feld ist, so kann daraus im Fall $j \neq j'$ nicht gefolgert werden, daß von $b_{i'}$ aus b_i nächstgelegenes mit m_j markiertes Feld ist. Diese Tatsache, daß ein Feld in diesem Sinne nächstgelegenes markiertes Feld für eine unbeschränkte Anzahl von Ausgansfeldern ist, ist auch der Grund dafür, daß sich die Speicherstruktur der TAS nicht durch einen Graphen im Sinne der "Graphen-Maschine" von A. Schönknecht ([Sch 62], neuere Darstellung in [Sch 74]) beschreiben läßt.

Durch Kombination von Satz 6.4 mit Satz 4.4 läßt sich aber schon eine erste Aussage zur Simulation einer allgemeinen TAS durch eine RAM herleiten:

Satz 6.5: Sei $\mathcal{T}$ eine 1-Band-TAS mit Bandbeschränkung $L(n)$ und Zeitbeschränkung $T(n)$, $L(n)$ sei RAM-gutartig. Dann gibt es eine zu $\mathcal{T}$ äquivalente RAM $\mathcal{R}$, die mit Zeitbeschränkung $C \cdot T(n) \cdot (\log L(n))^2$ arbeitet.

Beweis: $\mathcal{R}$ geht zur Simulation von $\mathcal{T}$ in der folgenden Weise vor:

1) $\mathcal{R}$ berechnet zunächst $L(n)$, danach $k := \lceil \log L(n) \rceil$ durch Berechnung von 2,4,8,... und Vergleich mit $L(n)$. $\mathcal{R}$ kommt dafür mit $C_1 \cdot L(n)$ Schritten aus.

2) Zu $\mathcal{T}$ gibt es nach Satz 4.4 eine äquivalente TAS $\mathcal{T}'$ in Normalform mit Parametern k (wie in 1) berechnet), s (s. unten) und $f=5$. $\mathcal{R}$ simuliert nun diese TAS $\mathcal{T}'$ und damit auch das Verhalten von $\mathcal{T}$.

Das Programm von $\mathcal{R}$ besteht also einerseits aus einem Programm zur Berechnung von k nach 1), andererseits aus dem Programm einer RAM $\mathcal{R}'$ zur Simulation von $\mathcal{T}'$, wobei nach Satz 4.4 die Spezifikation von $\mathcal{T}'$ aus der von $\mathcal{T}$ effektiv berechnet werden kann. Diese Berechnung von $\mathcal{T}'$ kann von $\mathcal{R}$ in einer Anzahl von Schritten vorgenommen werden, die proportional zur Anzahl der Marken von $\mathcal{T}'$ und damit zu $\log L(n)$ ist. Der Parameter s , den $\mathcal{T}'$ zur Simulation benötigt, kann vorweg aus den Parametern k' (Markenzahl) und s' (Größe des Bandalphabets) von $\mathcal{T}$ berechnet werden, $k = \lceil \log L(n) \rceil$ wird aus dem ersten Rechengang übernommen. Die Aussage von Satz 6.4 gilt aber auch in diesem Fall,

in dem die Anzahl k der Marken der TAS nicht von vornherein durch die Spezifikation von $\tilde{T}'$ gegeben ist, sondern in einem vorausgehenden Rechengang erst ermittelt wird. Überall dort, wo im Beweis zu Satz 6.4 bei fest gegebenem k aus der Spezifikation der gegebenen TAS in Normalform ein entsprechendes Teilprogramm der simulierenden RAM $\tilde{R}'$ zu konstruieren wäre (z.B. bei der Einrichtung der Registerblöcke der Länge $2.k$ im Adressenspeicher), kann $\tilde{R}$ ein entsprechendes "Unterprogramm" aufrufen, das das von zuvor berechnete k als Parameter verarbeitet und damit die Funktion des Teilprogramms von $\tilde{R}'$ übernimmt.

Da $\tilde{T}'$ eine TAS in Normalform ist, kommt $\tilde{R}$ nach Satz 6.4 zur Initialisierung der Registerorganisation mit $C_2.k.L'(n) = C_3.k.L(n)$ und zur eigentlichen Simulation mit $C_4.k.T'(n)$ Schritten aus, wenn $L'(n)$ die Bandbeschränkung und $T'(n)$ die Zeitbeschränkung von $\tilde{T}'$ ist. $\tilde{T}'$ simuliert andererseits $\tilde{T}$ mit Zeitbeschränkung $C_5.k.\sqrt[k]{L(n)}.T(n)$, nach Abschnitt 4 (S. 54/55) ist das für $k = \lceil \log L(n) \rceil$ durch $C_6.T(n). \log L(n)$ abschätzbar. Insgesamt ergibt sich damit als Zeitbeschränkung für $\tilde{R}$

$$C_1.L(n) + C_3.L(n). \log L(n) + C_4.C_6.T(n).(\log L(n))^2 \\ \leq C.T(n).(\log L(n))^2,$$

was zu beweisen war.

Das in Satz 6.5 erzielte Ergebnis kann noch verbessert werden, der verwendete Beweisgedanke ist recht umständlich: Es wird erst eine äquivalente TAS $\tilde{T}'$ mit verhältnismäßig vielen $(\log L(n))$ Marken konstruiert und diese TAS $\tilde{T}'$ dann durch eine RAM simuliert. Ein besseres Ergebnis läßt sich

durch direkte Simulation erzielen, wobei auch die zugrundeliegende Datenstruktur besser zum Ausdruck kommt: Dadurch, daß bei einer k-stufigen Indexhierarchie $k = \lceil \log L(n) \rceil$ gewählt wird, geht die indexsequentielle Speicherungsform in die eines binären Baumes ($l(n)$ Blätter, Höhe $\log L(n)$) über. Ein direkter, unabhängiger Beweis für die Simulation einer TAS durch eine RAM mit Zeitbeschränkung $C.T(n). \log T(n)$ (die Bandbeschränkung wird dort nicht explizit einbezogen) findet sich in [Fl 74]. Ein Beweis, in dem explizit ein binärer Baum zur Simulation assoziativer Zugriffe aufgebaut wird, soll auch hier im nächsten Abschnitt noch folgen, wobei die Beweisidee gleich auf das allgemeinere Modell der RAS (Random-Access-Maschine mit assoziativem Speicherzugriff) angewandt wird. Hier soll zunächst nur das Ergebnis (Satz 7.3) angeführt werden:

Zu jeder 1-Band-TAS mit k Marken, Zeitbeschränkung $T(n)$ und Bandbeschränkung $L(n)$ ($L(n)$ RAM-gutartig) gibt es eine äquivalente RAM mit Zeitbeschränkung $C.k.T(n). \log L(n)$.

Insgesamt lassen sich also die Beziehungen zwischen TAS und RAM in Kurzform so darstellen:

Korollar 6.6: Für die Komplexitätsklassen der TAS und der RAM gilt

$$C_{RAM}^1(T(n)) \subseteq C_{TAS}^{1,\infty}(T(n)^{1+\varepsilon}) \quad \text{für jedes } \varepsilon > 0,$$

$$C_{TAS}^{\omega,\infty}(L(n), T(n)) \subseteq C_{RAM}^1(T(n). \log L(n)).$$

Damit zeigt sich, daß trotz des verschiedenen Ursprungs der beiden Maschinenmodelle ihre Komplexitätsklassen doch recht nahe beieinanderliegen.

7. Random-Access-Maschinen mit assoziativem Speicherzugriff

Nach den bisherigen Betrachtungen liegt die Frage nahe, wie sich das Konzept des assoziativen Speicherzugriffs mit dem der Random-Access-Maschine verbinden läßt und welche Ergebnisse sich in Bezug auf die Komplexitätsbeschreibung durch solche Maschinenmodelle erzielen lassen. Die bisher einzige veröffentlichte Überlegung in dieser Richtung stammt von J.Hartmanis, der sich mit dem verwandten Modell der "Random Access Stored Program Machine" (RASP) beschäftigt hat.[Ha 71b]. (Daß jedes der beiden Modelle, das der RAM und das der RASP, jeweils das andere in Linearzeit simulieren kann, haben Cook und Reckhow gezeigt [CR 73] .) Hartmanis definiert als RASPA eine RASP mit der zusätzlichen Möglichkeit, auf Register in einem Schritt zuzugreifen, wenn nicht die Adresse, sondern ein Teil (Präfix) der gespeicherten Zahl gegeben ist; er beweist dafür den folgenden Satz:

Satz 7.1: Zu jeder RASPA mit Zeitbeschränkung $T(n)$ gibt es eine äquivalente ("normale") RASP mit Zeitbeschränkung $C.T(n)^2$.

Der Beweis ergibt sich in naheliegender Weise aus dem Konzept, daß der assoziative Zugriff jeweils durch eine sequentielle Suche in einer Liste der benutzten Register simuliert wird. Für nähere Einzelheiten der Definition und des Beweises sei auf [Ha 71b] verwiesen.

Hier soll nun das Modell der RAM - und zwar das der RAM mit einem eindimensionalen Speicher - durch Hinzunahme von

"assoziativen" Befehlen zur RAS (Random-Access-Maschine mit assoziativem Speicherzugriff) erweitert werden. Dazu sollen noch einmal die Befehle der RAM aufgeführt werden:

- | | |
|-----------------------------|--------------------------------|
| 1) $X_i \leftarrow X_j$ | 5) $X_i \leftarrow X_i \div 1$ |
| 2) $X_i \leftarrow R_{X_j}$ | 6) TRA m if $X_j > 0$ |
| 3) $R_{X_i} \leftarrow X_j$ | 7) Read X_i |
| 4) $X_i \leftarrow X_i + 1$ | 8) Print X_i |

Dazu treten für die RAS die Befehle

- 9) $X_i \leftarrow \text{ASSL}_{X_j}$
- 10) $X_i \leftarrow \text{ASSR}_{X_j}$
- 11) $\text{ASSL}_{X_i} \leftarrow X_j$
- 12) $\text{ASSR}_{X_i} \leftarrow X_j$

Um die Wirkung der Befehle 9) - 12) erläutern zu können, muß zuvor noch die Wirkungsweise der Befehle 2) und 3) genauer festgelegt werden: Die RAM bzw. RAS verfüge über ein Adreßregister AR, in das bei der Ausführung des Befehls $X_i \leftarrow R_{X_j}$ zunächst der Inhalt von Register j gebracht wird, anschließend wird $X_i \leftarrow X_{AR}$ ausgeführt. Jede Ausführung eines Transportbefehls mit indirekter Adressierung besetzt also das Adreßregister neu. Dadurch ist auch bei der RAM bzw. RAS etwas zur Kopfstellung der TM und der TAS Analoges gegeben: So wie dort die Stellung des Lesekopfes die Stelle angibt, an der Daten vom Speicherband geholt oder zu der Daten gebracht werden, gibt bei der RAM bzw. RAS das Adreßregister die Stelle im indirekt adressierbaren Speicher an, auf die jeweils zugegriffen wird.

Dies vorausgesetzt, bewirkt eine Ausführung von Befehl 9) folgendes: In das Register i wird die Adresse desjenigen indirekt adressierbaren Registers r gebracht, dessen Inhalt mit X_j übereinstimmt und das außerdem (es kann ja mehrere solche Register geben) von dem Register aus, dessen Adresse im Adreßregister steht, das nächstgelegene Register "nach links" ist. Der Befehl bewirkt also einen Transport

$$X_i \leftarrow r \quad \text{mit } r = \max (s \mid s \leq X_{AR} \text{ und } R_s = X_j).$$

Entsprechend bewirkt Befehl 10) einen Transport

$$X_i \leftarrow r \quad \text{mit } r = \min (s \mid s \geq X_{AR} \text{ und } R_s = X_j).$$

Analog seien die Operationen 11) und 12) definiert. Wenn in dem entsprechenden Speicherbereich kein Register R_s mit der Eigenschaft $R_s = X_j$ existiert, so stoppt die Maschine. Auch für die assoziativen Transportbefehle soll gelten, daß durch ihre Ausführung das Adreßregister jeweils neu (nun mit der Adresse r) besetzt wird; dies entspricht der zuvor erläuterten Analogie zwischen Lesekopf und Adreßregister. Daß durch die assoziative Zugriffsoperation jeweils eine Adresse und nicht ein Registerinhalt ermittelt und transportiert wird, erscheint sinnvoll, da der Inhalt des angesprochenen Registers hier ja vor Ausführung der Operation schon bekannt ist.

Wie schon beim Modell der TAS erwähnt, weicht die Bezeichnung dieser Befehle als "assoziativ" etwas von der üblichen Auffassung in der mehr technisch orientierten Literatur über assoziative Speicher ab: Während dort angenommen wird, daß eine Operation in allen Registern mit einem bestimmten Inhalt gleichzeitig vorgenommen wird,

erfolgt hier der Zugriff nur auf jeweils ein Register (von dem aus dann u.U. zum nächsten "gesprungen" werden kann). Gemeinsam ist jedoch, daß die Steuerung des Zugriffs durch den Inhalt des Registers erfolgt.

Diese Definition der RAS ist insofern analog zur Definition der TAS - und auch in dieser Hinsicht abweichend von der technisch orientierten Literatur über assoziative Speicher -, als ein assoziativer Speicherzugriff nicht nur vom jeweiligen Inhalt der Register, sondern auch von der Lage des Registers, auf das zuletzt ein Zugriff erfolgte, abhängt. Eine andere Möglichkeit der Definition wäre, daß ein assoziativer Speicherzugriff immer auf das jeweils erste Register mit einem bestimmten Inhalt erfolgt, etwa entsprechend der Definition der 1TAS. Die Ergebnisse bei der TAS zeigen, daß zumindest für den Spezialfall der On-Line-Spracherkennung dort das zweite Modell wesentlich schwächer ist als das erste. Eine RAS mit einem "adreßregister-unabhängigen" assoziativen Zugriff auf das jeweils erste Register mit einem bestimmten Inhalt läßt sich jedoch natürlich leicht durch das hier angegebene Modell der RAS simulieren, indem nach jedem Speicherzugriff, der das Adreßregister ändert, eine Operation eingefügt wird, die das Adreßregister wieder auf 1 setzt, und indem nur mit den Operationen mit "ASSR" gearbeitet wird.

In der Fähigkeit zum inhaltsgesteuerten Zugriff geht die hier definierte RAS dadurch über die TAS wesentlich hinaus, daß hier eine beliebige natürliche Zahl als "Ziel" den Zugriff steuern kann; die RAS entspricht in dieser

Hinsicht also einer TAS mit unendlich vielen Marken.

Im folgenden soll gezeigt werden, wie sich eine RAS durch eine normale RAM simulieren läßt.

Satz 7.2: Zu jeder RAS $\mathcal{A}$ mit Zeitbeschränkung $T(n)$

($T(n)$ RAM-gutartig) gibt es eine äquivalente RAM $\mathcal{R}$ mit Zeitbeschränkung $C \cdot T(n) \cdot (\log T(n))^2$.

Beweis: Es wird zunächst eine zu $\mathcal{A}$ äquivalente RAM $\mathcal{R}$ mit mehreren Speichern, von denen einer zweidimensional ist, konstruiert, anschließend werden Lemma 6.2 und 6.1 angewandt. Der Grundgedanke der Simulation ist dabei, daß sich $\mathcal{R}$ durch eine geeignete Buchführung merkt, wo im Speicher von $\mathcal{A}$ welche Zahlen (jede Zahl kann ja bei der RAS die Funktion einer "Marke" haben, in diesem Sinne werden im folgenden die gespeicherten Zahlen manchmal auch als Marken bezeichnet) auftreten; es werden also die Beweisgedanken von Satz 4.4 und Satz 6.3 miteinander verbunden. $\mathcal{R}$ baut sich dabei in gewisser Weise eine binäre Baumstruktur auf: Jedes 2^{i-1} -te Register im Arbeitsspeicher (in dem die normalen Operationen von $\mathcal{A}$ simuliert werden) werde als "Register i-ter Stufe" bezeichnet (entsprechend den Referenzfeldern bei Satz 4.1 und 4.4), ihm sind in einem zweidimensionalen Markenspeicher "Registerblöcke i-ter Stufe" ($i \leq L := \lceil \log T(n) \rceil$) zugeordnet. Für jede potentielle "Marke" m (also für jede Zahl zwischen 1 und $T(n)$) und für jedes Register A_j (es sei ein Register i-ter Stufe) ist im Markenspeicher in einem Block i-ter Stufe gespeichert, wie oft die Zahl m in den $2^{i-1}, 2^{i-2}, \dots, 2, 1$ Registern zwischen A_j und den nächstgelegenen Registern A_j, i -ter, $(i-1)$ -ter, ..

.., zweiter, erster Stufe vorkommt. Ein solcher Registerblock enthält dazu $2.L + C$ Register ¹⁾, für jede Marke gibt es $T(n)$ Registerblöcke. Die $(2.L + C) \cdot T(n)$ Blöcke zur Marke m bilden die m -te Zeile des Markenspeichers, er ist ein zweidimensionaler Speicher mit $(2.L + C) \cdot T(n)^2$ Registern. Nur ein kleiner Teil dieser Register wird während der Simulation von A durch R tatsächlich genutzt, beim Rest bleibt die Null, die nach Definition der RAM zu Beginn einer Rechnung den Inhalt jedes Registers bildet, unverändert stehen.

Die in den Blöcken des Markenspeichers enthaltene Information gestattet es, zu gegebenem m die jeweils nächstgelegene "Marke" m aufzufinden (durch Überprüfen der entsprechenden Register des Markenspeichers auf positiven Inhalt, das genaue Verfahren wird später erläutert). Ein Adressenspeicher dient zum schnellen Zugriff auf die Register des Markenspeichers, ein Adreßverweisspeicher wiederum zum Zugriff auf den Adressenspeicher, wenn die Adresse eines Registers im Arbeitsspeicher gegeben ist. Die Simulation von Sprüngen mit assoziativem Speicherzugriff kann damit in ähnlicher Weise wie im Beweis zu Satz 4.1 bzw. 4.4 vor sich gehen; allerdings ist die Information, die dort in einem Band der TAS (auf Symbol- und Markenspur verteilt) gespeichert werden konnte (Symbole $(\overset{m}{L}), (\overset{m}{R})$ einerseits, Marken $m'_1, \dots, m'_k$ andererseits), hier auf zwei Speicher,

1) Es wäre möglich, hier Speicherplatz zu sparen und für Blöcke i -ter Stufe jeweils nur $2.i + C$ Register vorzusehen; da es hier aber nicht auf optimale Speicherausnutzung, sondern nur auf die Zugriffszeit ankommt, werden für jeden Block $2.L + C$ Register (aus der genaueren Beschreibung ergibt sich: $C = 4$) vorgesehen.

Markenspeicher und Adressenspeicher, verteilt. Auch unter einem anderen Aspekt ergibt sich ein Unterschied zur TAS: Dort erfolgten die Sprünge von einem Referenzfeld zum anderen mit Hilfe von Marken, die Sprünge wurden also in gewisser Weise vom Sprungziel her (Marke, auf die gesprungen wird) gesteuert. Bei der RAM dagegen muß bei einem Sprung von einem Registerblock im Adressenspeicher zu einem anderen die Angabe, wohin gesprungen werden soll, als Adresse im Ausgangsblock gegeben sein. Auch deshalb sind im Adressenspeicher $2.L+C$ Register je Registerblock nötig, außerdem gestattet diese Organisation (die im einzelnen später erläutert wird) eine einfache Beziehung zwischen Zieladressen (im Adressenspeicher) und Information über die Anzahl der Marken im übersprungenen Speicherbereich (im Markenspeicher).

$\bar{R}$ verfügt damit über die folgenden Speicher:

- einen Arbeitsspeicher A zur Simulation der "normalen" Operationen von $\bar{A}$ ($T(n)$ Register),
- einen zweidimensionalen Markenspeicher M, bestehend aus $T(n)$ Zeilen mit je $(2.L+C).T(n)$ Registern,
- einen Adressenspeicher mit $(2.L+C).T(n)$ Registern (je einem Register für jede Spalte des Markenspeichers),
- einen Adreßverweisspeicher mit $T(n)$ Registern (je eines für jeden Block des Adressenspeichers).

Die Beziehungen der einzelnen Speicher von $\bar{A}$ und $\bar{R}$ zueinander läßt sich durch Abb. 19 beschreiben, die genauere Organisation der Register von Adressen- und Markenspeicher durch Abb. 20 und 21.

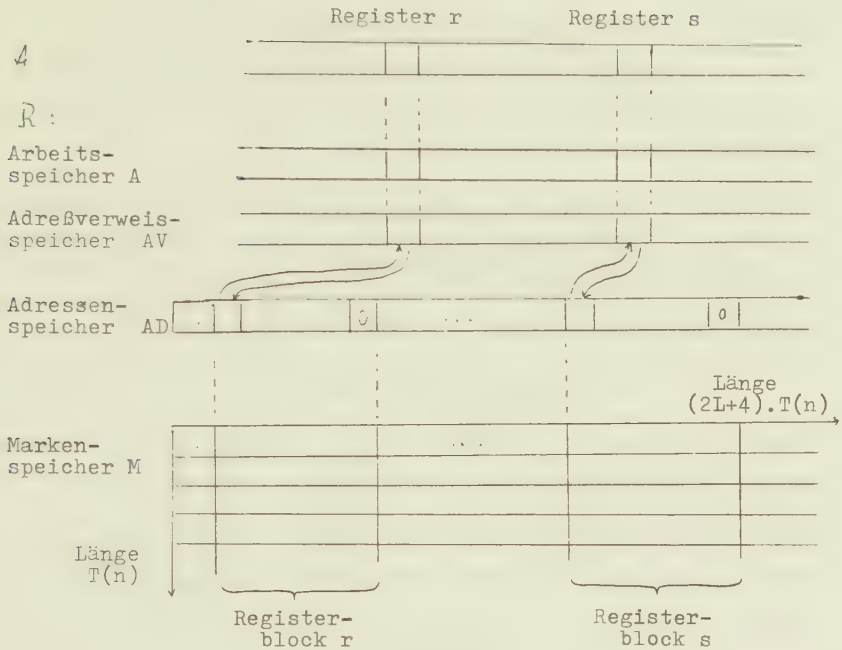


Abb.19: Speicher von $\hat{A}$ und $\hat{A}$

Im einzelnen gilt für die Register des r -ten Registerblocks im Adressenspeicher, wenn es sich um einen Block i -ter Stufe handelt:

- $AD_r^{(-1)}$ enthält die Zahl r (d.h. die zugehörige Adresse für Arbeits- und Adreßverweisspeicher),
- $AD_r^{(0)}$ enthält die Zahl 0,
- $AD_r^{(j)}$ ($1 \leq j \leq i$) enthält die Adresse von $AD_{r'}^{(j)}$, wobei $r' = r - 2^{j-1}$ ist, also die Adresse des j -ten Registers im nächsten Block j -ter Stufe nach links,
- $AD_r^{(j)}$ ($i+1 \leq j \leq L$) enthält eine 0,

$AD_r^{(L+j)}$ ($1 \leq j \leq i$) enthält die Adresse von $AD_{r'}^{(L+j)}$, wobei $r' = r + 2^{j-1}$ ist, also die Adresse des j -ten Registers im nächsten Block j -ter Stufe nach rechts,

$AD_r^{(L+j)}$ ($i+1 \leq j \leq L$) enthält eine 0.

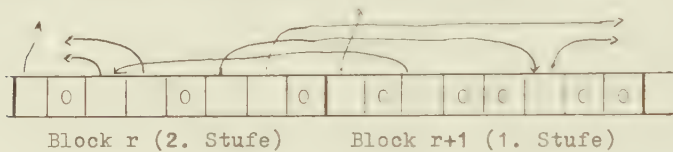


Abb. 20: Adressenspeicher (Fall $L = 3$)

Den Registern $AD_r^{(1)}, \dots, AD_r^{(i)}, AD_r^{(i+1)}, \dots, AD_r^{(i+i+1)}$ eines Blocks im Adressenspeicher entsprechen Register mit derselben (absoluten) Adresse in jeder Zeile m des Markenspeichers, wobei für diese "Markenregister" $M_{r,m}^{(j)}$ gilt:

$M_{r,m}^{(j)}$ ($1 \leq j \leq i$) enthält die Zahl p , wenn bei 4 (und damit auch im Arbeitsspeicher von K) die Zahl (Marke) m insgesamt p -mal in den Registern $A_r, A_{r-1}, \dots, A_{r-2^{j-1}+1}$ vorkommt (Das r -te Register sei dabei ein Register mindestens j -ter Stufe).

$M_{r,m}^{(L+j)}$ ($1 \leq j \leq i$) enthält die Zahl q , wenn bei 4 die Zahl (Marke) m insgesamt q -mal in den Registern $A_r, A_{r+1}, \dots, A_{r+2^{j-1}-1}$ vorkommt.

Für $i+1 \leq j \leq L$ kann der Inhalt von $M_{r,m}^{(j)}$ und $M_{r,m}^{(L+j)}$ beliebig sein, etwa 0; er wird nicht benutzt.

Durch die in den einzelnen Registern des Markenspeichers gespeicherten Zahlen wird also für jede Marke angegeben, wie oft sie im Arbeitsspeicher in dem Speicherbereich vorkommt, der gerade (indirekt) durch die entsprechende Adresse im gleichen Register des Adressenspeichers gekennzeichnet ist.

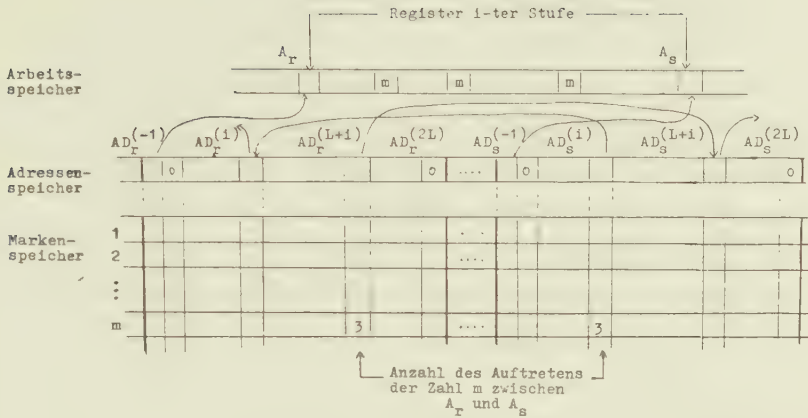


Abb.21: Markenspeicher und Adressenspeicher

Aus dieser Organisation und aus der Analogie zum Beweis bei der TAS wird schon erkennbar, wie die Arbeitsweise von $\bar{R}$ bei der Simulation von $\bar{A}$ ist. Der Vollständigkeit halber soll aber nun noch genauer beschrieben werden, wie $\bar{R}$ die folgenden Aufgaben löst und wieviel Schritte dazu jeweils ausreichend sind:

- 1) Einrichten der Registerorganisation im Adressenspeicher (Initialisierung)
- 2) Simulation des assoziativen Zugriffs auf ein Register (Suche nach der nächstgelegenen Marke)
- 3) Berichtigen der Zählregister im Markenspeicher bei Änderungen im Arbeitsspeicher ("Setzen", "Löschen" von Marken).

1) Initialisierung:

Da nach Voraussetzung zu gegebenem n die Funktion $T(n)$ in $C_1 \cdot T(n)$ Schritten berechenbar ist, kann $\bar{R}$ in $C_2 \cdot T(n)$

Schritten auch $L := \lceil \log T(n) \rceil$ berechnen. Damit können

- in $C_3 \cdot T(n)$ Schritten die Blöcke im Adressenspeicher "abgesteckt" und die Adressenverweise erster Stufe eingetragen werden. Es werden also die Register $-1, 0, 1$ und $L+1$ jedes Blockes richtig besetzt, in allen anderen Registern steht vorläufig eine Null.
- in $C_4 \cdot \frac{L}{I}$ Schritten die Register i und $L+i$ ($i = 2, \dots, L$) der Blöcke i -ter Stufe besetzt werden, vorausgesetzt, daß die Adressenverweise auf die Blöcke i -ter Stufe jeweils schon richtig eingetragen sind.

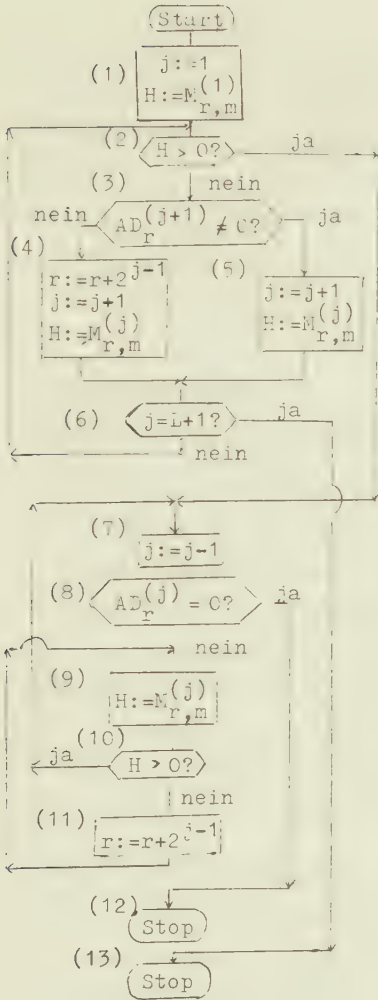
Beim ersten "Durchgang" werden auch die Adressen der jeweils zugehörigen Blöcke des Adressenspeichers im Adreßverweisregister eingetragen; Eintragungen im Arbeitsspeicher werden erst während der eigentlichen Simulation vorgenommen. Ebenso ist keine Initialisierung des Markenspeichers nötig, da nach Voraussetzung zu Beginn der Rechnung einer RAM alle Register mit der Zahl 0 besetzt sind; erst während der eigentlichen Simulation werden hier andere Zahlen eingetragen. Damit kann sich, obwohl $\bar{R}$ im Verlauf der Rechnung bis zu $O(L \cdot T(n))^2$ Register zur Verfügung hat, die Initialisierung auf Adressen- und Adreßverweisregister beschränken; $\bar{R}$ kommt dafür mit

$$C_2 \cdot T(n) + C_3 \cdot L \cdot T(n) + \sum_{i=2}^L C_4 \cdot \frac{L}{I} \leq C_5 \cdot L \cdot T(n)$$

Schritten aus.

2) Suche nach Marken (assoziativer Zugriff):

Es sei z.B. die Operation $ASSL_{X_i} \leftarrow X_j$ von $\mathcal{A}$ zu simulieren. Dabei kommt es für $\bar{R}$ darauf an, die Adresse des Registers im Arbeitsspeicher zu finden, in dem die Zahl ("Marke")



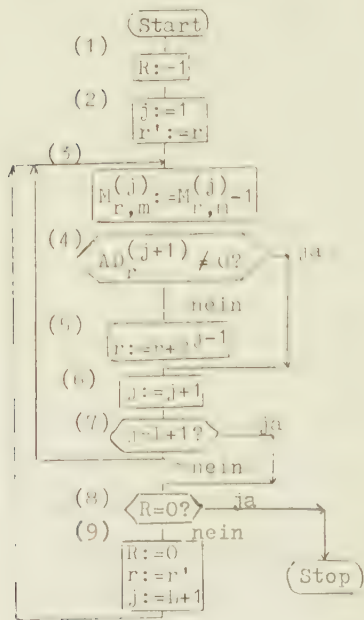
- (2) Frage nach Marke m im entsprechenden Speicherabschnitt
- (3) Steht im Register $AD_r^{(j+1)}$ noch eine Adresse (Block höherer Stufe)?
- (4),(5) Neuer Versuch mit größerem Speicherabschnitt
- (4) Adresse von Register $AD_r^{(j+1)}$ steht im $r+2^{j-1}$ Register $AD_r^{(j+1)}$
- (6) Gibt es keinen Block höherer Stufe mehr? (erkennbar an der 0 im Register $AD_r^{(j)}$)
- (8) Ist wieder Stufe 1 erreicht?
- (9) Suche in kleinerem Speicherabschnitt
- (11) Suche im nächsten Speicherabschnitt gleicher Stufe. Adresse von Register $AD_r^{(j)}$ steht im $r+2^{j-1}$ Register $AD_r^{(j)}$
- (12) Gesuchte Adresse = Inhalt von $AD_r^{(-1)}$
- (13) Abbruch, keine Marke m vorhanden

$m = X_j$ gespeichert ist und das außerdem dem Register, dessen Adresse durch $r = X_{AR}$ gegeben ist, von "links" her am nächsten gelegen ist. Dies ist durch den oben angegebenen Algorithmus möglich, durch den zunächst für immer größere Bereiche $(1, 2, 4, \dots, 2^i \text{ Register, } i \leq L-1)$, dann für wieder

kleiner werdende Bereiche ($2^{i-1}, 2^{i-2}, \dots, 2, 1$ Register) des Arbeitsspeichers überprüft wird, ob dort die Marke m vorhanden ist (Ein Register H dient dabei als Hilfsregister). Jeder der Teile (1) - (11) des Algorithmus erfordert nur eine konstante Anzahl von Schritten, und die Folgen (2) - (5) und (7) - (11) werden höchstens je L -mal durchlaufen; es sind also zusammen nicht mehr als $C_6 \cdot L$ Schritte nötig.

3) Setzen bzw. Löschen von Marken:

Wird von $\star$ in einem der indirekt adressierbaren Register eine Zahl m eingetragen - auf die später als Marke assoziativ zugegriffen werden könnte -, so sind bei $\star$ alle 2.1 Eintragungen in den Registern der m -ten Teile des Markenspeichers zu berichtigen, die auf dieses Register Bezug nehmen. Dies geschieht durch den folgenden Algorithmus:



- (1) $R = 1$: Richtung nach links
 $R = 0$: Richtung nach rechts
- (2) Beginn auf Stufe 1
Aufbewahren des Wertes von r
- (3) Markenzähler wird herabgesetzt
- (4) Steht im Register $AD_r^{(j+1)}$ noch eine Adresse (Block höherer Stufe)?
- (5) Aufsuchen des nächsten Blocks (der dann ein Block höherer Stufe ist)
- (6) Gibt es keinen Block höherer Stufe mehr? (erkennbar an der 0 im Register $AD_r^{(j)}$)
- (7) Wechsel der Richtung, Wiederherstellen von r

Der Algorithmus beschreibt die Operationen von $\bar{R}$, wenn von A im Register r die Zahl m überschrieben wird.

Ganz analog wird in den Zählregistern des Markenspeichers die Eintragung einer Zahl (Marke) m registriert, dann wird in Schritt (3) des Algorithmus jeweils eine 1 addiert.

$\bar{R}$ benötigt also auch für das Setzen und Löschen von Marken nicht mehr als $C_7 \cdot L$ Schritte.

Aus 1) - 3) folgt insgesamt, daß A von einer RAM $\bar{R}$ (mit einem zweidimensionalen Speicher) mit Zeitbeschränkung $C_8 \cdot T(n) \cdot \log T(n)$ simuliert werden kann. Nach Lemma 6.2 gibt es aber zu jeder zweidimensionalen RAM mit Zeitbeschränkung $T(n)$ eine äquivalente eindimensionale RAM mit Zeitbeschränkung $T(n) \cdot \log T(n)$, außerdem können mehrere Speicher auf einen Speicher reduziert werden, wobei sich die Zeitbeschränkung nur um einen konstanten Faktor vergrößert (Lemma 6.1). A kann daher von einer eindimensionalen RAM mit nur einem Speicher mit Zeitbeschränkung $C_9 \cdot (T(n) \cdot \log T(n)) \cdot \log (T(n) \cdot \log T(n)) \leq C \cdot T(n) \cdot (\log T(n))^2$ simuliert werden, was zu beweisen war.

Das in diesem Beweis entwickelte Verfahren zur Simulation assoziativer Zugriffe läßt sich nun auch sehr leicht auf die Simulation einer TAS durch eine RAM anwenden, wobei man das in Abschnitt 6 (S. 91) schon angekündigte Ergebnis erhält:

Satz 7.3: Zu jeder 1-Band-TAS $\bar{T}$ mit k Marken, die mit Zeitbeschränkung $T(n)$ und mit Bandbeschränkung $L(n)$ arbeitet ($L(n)$ RAM-gutartig), gibt es eine äquivalente RAM $\bar{R}$ mit Zeitbeschränkung $C \cdot k \cdot T(n) \cdot \log L(n)$.

Beweis: Der Beweisgedanke von Satz 7.2 kann, mit einer wichtigen Vereinfachung, übernommen werden: Die Register des Arbeitsspeichers der simulierenden AM entsprechen hier den $L(n)$ benutzten Bandfeldern von T . Daß die normalen TM-Operationen von T jeweils von einer RAM in einer konstanten Anzahl von Schritten simuliert werden können, wurde schon im Beweis zu Satz 6.4 (S. 84) erläutert. Sprünge der TAS T auf eine Marke lassen sich nach demselben Verfahren, wie im Beweis zu Satz 7.2 geschildert, mit Hilfe von Markenspeicher, Adressenspeicher und Adreßverweisspeicher simulieren, ebenso das Setzen und Löschen von Marken. Da jedoch hier T nur über beschränkt viele, nämlich k Marken verfügt, ist kein zweidimensionaler Markenspeicher nötig, sondern es genügen k lineare Markenspeicher (einer für jede Marke). R kann damit zunächst als RAM mit $k+3$ linearen Speichern konstruiert werden, die mit Zeitbeschränkung $C'.T(n).log L(n)$ arbeitet. Dazu gibt es wiederum nach Lemma 6.1 eine äquivalente RAM mit nur einem Speicher und Zeitbeschränkung $C.k.T(n).log L(n)$.

Abschließend soll noch eine andere Variante der RAS betrachtet werden: Bei dem bisher verwendeten Modell der RAS bezog sich der assoziative Zugriff immer auf den gesamten Inhalt eines Registers. Im Sinne der Unterscheidung von Abschnitt 1 stellt also jeweils ein ganzer Registerinhalt das Schlüsselwort dar, die Information steckt in der Adresse oder (indirekt) im Inhalt benachbarter Register. Man kann sich aber auch Situationen vorstellen, in denen es nützlich wäre, einen Befehl zur Verfügung zu haben, der folgendes bewirkt: Für eine beliebige Zahl (Inhalt eines

Registers X_j), die sich in der Dualdarstellung als Zeichenkette $v \in 1 \cdot \{0,1\}^*$ auffassen läßt, wird die Adresse des Registers gebracht, dessen Inhalt mit den ersten Bits (außer evtl. führenden Nullen) von v übereinstimmt. Dabei werden Registerinhalte, wie üblich, als Dualzahlen mit der höchsten Binärstelle auf der linken Seite aufgefaßt. Dieses Konzept liegt auch der assoziativen RASP in [Ha 71b] zugrunde. Sei dementsprechend eine PRAS (Präfix-RAS) definiert als eine RAM mit den folgenden zusätzlichen Operationen:

$$X_i \leftarrow \text{PASSL}_{X_j}$$

$$X_i \leftarrow \text{PASSR}_{X_j}$$

$$\text{PASSL}_{X_i} \leftarrow X_j$$

$$\text{PASSR}_{X_i} \leftarrow X_j$$

Dabei soll etwa $X_i \leftarrow \text{PASSL}_{X_j}$ das folgende bewirken:

$$X_i \leftarrow r \quad \text{mit } r = \max (s \mid s = X_{AR} \text{ und es gibt ein } k, \text{ so daß} \\ R_s = 2^k \cdot X_j + z_k \text{ mit } z_k < 2^k) ,$$

analog seien die anderen Operationen definiert.

Im folgenden soll zunächst gezeigt werden, wie sich eine PRAS - auch wenn nur +1 und -1 als arithmetische Operationen zur Verfügung stehen - durch eine RAS simulieren läßt, wobei auch hier die Zeitbeschränkung nur um einen logarithmischen Faktor anwächst. Durch Kombination dieses Ergebnisses mit dem vorigen läßt sich dann eine Aussage über die Simulation einer PRAS durch eine RAM herleiten.

Satz 7.4: Zu jeder PRAS $\mathcal{P}$ mit Zeitbeschränkung $T(n)$

($T(n)$ RAM-gutartig) gibt es eine äquivalente RAS $\mathcal{A}$ mit Zeitbeschränkung $T(n) \cdot \log T(n)$.

Beweis: $\mathcal{A}$ geht zur Simulation von $\mathcal{P}$ in der folgenden Weise vor:

1) $\mathcal{A}$ berechnet zunächst $T(n)$ und $L := \lceil \log (T(n)+1) \rceil$, die dabei als Zwischenergebnis anfallenden Zahlen $1, 2, 4, \dots, 2^L$ werden in L weiteren "Zählregistern" $Z_1, \dots, Z_L$ gespeichert. Dann werden für jede Zahl $z \in \{1, 2, \dots, T(n)\}$ die Menge aller möglichen Präfixe z' (d.h. aller Zahlen z' mit $z = 2^k \cdot z' + z_k$ für ein $z_k < 2^k$) nach dem folgenden Prinzip berechnet: Seien für eine Zahl z die Präfixe schon berechnet, sie seien $z = z_0 > z_1 > \dots > z_k = 1$. z habe in den letzten j Binärstellen eine 1, die $(j+1)$ -te Stelle sei eine 0. Dann hat $z+1$ als Präfixe die Zahlen $z_0+1, \dots, z_j+1, z_{j+1}, \dots, z_k$ (für $j = \log (z+1)$ erhöht sich dabei jeweils die Anzahl der Präfixe um 1). Werden nun nacheinander zu den Zahlen $1, 2, 3, \dots, T(n)$ die Präfixe berechnet, so hat jeweils jede 2^{i+1} -te Zahl ($i = 1, 2, 3, \dots, L-1$) in den letzten i Binärstellen eine 1 und in der $(i+1)$ -ten Binärstelle eine 0. Durch Mitzählen in den Zählregistern kann also $\mathcal{A}$ jeweils aus den k Präfixen von z in $C_1 \cdot L$ Schritten die Präfixe von $z+1$ berechnen. Sie werden dann in jeweils bis zu L Registern gespeichert, durch eine Adreßverweis-Organisation (wie im Beweis zu Satz 7.1) kann $\mathcal{A}$ dann bei gegebenem z auf die Präfixe von z unmittelbar zugreifen (Abb. 22). $\mathcal{A}$ kann also in $C_1 \cdot T(n) \cdot L$ Schritten alle Präfixe der Zahlen $1, \dots, T(n)$ berechnen und geeignet speichern.

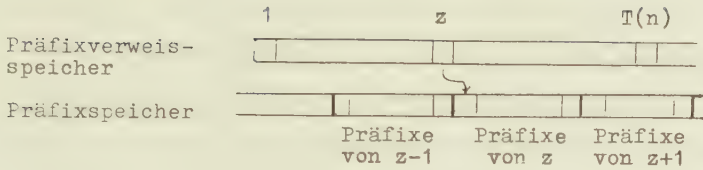


Abb.22: Speicherung aller Präfixe

2) Simulation der einzelnen Schritte von $\mathcal{P}$:

Der Arbeitsspeicher von $\mathcal{A}$ sei so angelegt, daß jedem Register des indirekt adressierbaren Speichers von $\mathcal{P}$ ein Block von L Registern bei $\mathcal{A}$ entspricht.

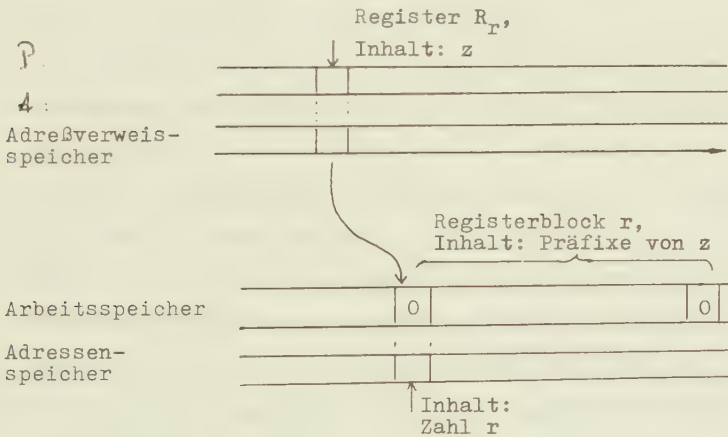


Abb.23: Simulation von $\mathcal{P}$ durch $\mathcal{A}$

Wird im r -ten Register R_r von $\mathcal{P}$ eine Zahl z eingetragen, so trägt $\mathcal{A}$ im Arbeitsspeicher im r -ten Registerblock nacheinander alle Präfixe von z ein, aus dem Präfixspeicher können diese in $C_2 \cdot L$ Schritten übernommen werden. Durch die sequentielle Anordnung der Registerblöcke im Arbeitsspeicher von $\mathcal{A}$ ist gewährleistet, daß für jede

Zahl z und jede Adresse r ($1 \leq z, r \leq T(n)$) gilt:

$$\max (s \mid s \leq r \text{ und } R_s = 2^k \cdot z + z_k \text{ mit } z_k < 2^k)$$

(R_s = s-tes Register im Arbeitsspeicher von $\tilde{P}$)

$$= \max (s \mid s \leq (l+2) \cdot r, \text{ und } z \text{ ist Inhalt eines Registers im } s\text{-ten Registerblock von } \mathcal{A})$$

Eine entsprechende Aussage gilt auch für $\min (s \mid \dots)$, also für Sprünge nach rechts.

Eine Operation $X_i \leftarrow \text{PASSL}_{X_j}$

von $\tilde{P}$ kann damit durch eine Operation

$$X_i \leftarrow \text{ASSL}_{X_j}$$

im Arbeitsspeicher von $\mathcal{A}$ und weitere anschließende Schritte von $\mathcal{A}$ (un von gefundenen Prefix aus die Nummer r des Registerblocks, die der Adresse r bei $\tilde{P}$ entspricht, zu finden), deren Anzahl durch $2 \cdot l$ beschränkt ist, simuliert werden. Analog werden die anderen assoziativen Zugriffsoperationen von $\tilde{P}$ simuliert.

Insgesamt kann damit $\tilde{P}$ durch eine RAS $\mathcal{A}$ mit mehreren Speichern, von denen einer assoziativen Zugriff gestattet, mit Zeitbeschränkung $T(n) \cdot \log T(n)$ simuliert werden.

Eine Kombination von Satz 7.2 und 7.4 ergibt nun:

Satz 7.5: Zu jeder PRAS $\tilde{P}$ mit Zeitbeschränkung $T(n)$

($T(n)$ AS-gutartig) gibt es eine äquivalente RAM $\tilde{\lambda}$ mit Zeitbeschränkung $C \cdot T(n) \cdot (\log T(n))^3$.

Beweis: Nach Satz 7.4 läßt sich $\tilde{P}$ durch eine RAS $\mathcal{A}$ mit Zeitbeschränkung $C_1 \cdot T(n) \cdot \log T(n)$ simulieren. Eine kleine technische Schwierigkeit liegt noch darin, daß $\mathcal{A}$ eine RAS mit mehr als einem Speicher ist. Satz 7.2 gilt aber offen-

sichtlich auch für den Fall, daß $\mathcal{A}$ mehrere Speicher hat, von denen einer assoziativ zugreifbar ist. $\mathcal{A}$ kann daher wiederum durch eine RAM $\mathcal{R}$ simuliert werden, nach Lemma 6.1 kann $\mathcal{R}$ als RAM mit nur einem Speicher gewählt werden. Insgesamt arbeitet damit $\mathcal{R}$ mit Zeitbeschränkung

$$C' \cdot C_1 \cdot T(n) \cdot \log T(n) \cdot (\log (T(n) \cdot \log T(n)))^2 \\ \leq C \cdot T(n) \cdot (\log T(n))^3$$

für eine geeignete Konstante C , was zu beweisen war.

Der Vollständigkeit halber soll noch auf zwei andere Beziehungen zwischen Maschinenmodellen eingegangen werden, zwischen RAS und TAS und zwischen PRAS und RAS (Simulation der RAS durch die PRAS).

RAS und TAS: Einerseits ist es nicht schwer, zu sehen, daß, ähnlich wie eine TM sich durch eine RAM in Linearzeit simulieren läßt, auch eine Simulation einer TAS durch eine RAS in Linearzeit möglich ist. Da die RAS über beliebig viele assoziativ zugreifbare "Marken" verfügt, kommt es auf die Anzahl der Marken der TAS nicht an; durch das Adreßregister wiederum läßt sich die Funktion des Lesekopfes bei assoziativen Sprüngen nachbilden. Andererseits kann man umgekehrt durch Kombination von Satz 7.5 und Satz 6.4 zeigen, daß es zu jedem $k \in \mathbb{N}$ und zu jeder RAS oder PRAS $\mathcal{A}$, die mit Zeitbeschränkung $T(n)$ arbeitet, eine äquivalente TAS $\mathcal{T}$ mit k Marken gibt, die für jedes $\varepsilon > \frac{1}{k}$ mit Zeitbeschränkung $T(n)^{1+\varepsilon}$ arbeitet. Denn zu $\mathcal{A}$ gibt es eine äquivalente RAM $\mathcal{R}$ mit Zeitbeschränkung $T'(n) = T(n) \cdot (\log T(n))^3$, dazu wieder eine äquivalente TAS mit

Zeitbeschränkung $T'(n)^{1+\frac{1}{k}} = T(n)^{1+\frac{1}{k}} \cdot (\log T(n))^{3 \cdot (1+\frac{1}{k})}$
 $< T(n)^{1+\varepsilon}$ für $\varepsilon > \frac{1}{k}$ (n genügend groß).

PRAS und RAS: Die Simulation einer PRAS durch eine RAS wurde in Satz 7.4 behandelt. Auch umgekehrt ist eine Simulation nicht ganz trivial, denn bei der RAS erfordert ein assoziativer Zugriff volle Übereinstimmung zwischen Suchargument und Registerinhalt, bei der PRAS dagegen nur partielle Übereinstimmung. Bei einer direkten Simulation einer RAS durch eine PRAS könnte also ein assoziativer Sprung "zu früh" enden. Trotzdem läßt sich zu jeder RAS $\mathcal{A}$ mit Zeitbeschränkung $T(n)$ eine äquivalente PRAS $\mathcal{B}$ mit Zeitbeschränkung $O(T(n))$ konstruieren. Dies kann dadurch verdeutlicht werden, daß $\mathcal{B}$ zunächst als PRAS mit zwei Speichern konstruiert wird. Ein Speicher dient normal als Arbeitsspeicher, ein anderer als Assoziativspeicher. Ist im Arbeitsspeicher im r -ten Register die Zahl z gespeichert, so speichert das r -te Register im Assoziativspeicher die Zahl $T+z$ ($T := 2^{\lceil \log T(n) \rceil}$). Ein assoziativer Zugriff bei auf eine Zahl z kann dann durch einen prefix-assoziativen Zugriff auf $T+z$ simuliert werden. Da dabei alle im Assoziativspeicher gespeicherten Zahlen und alle Suchargumente gleiche Länge ($\log T + 1$) haben, gibt es keine Schwierigkeiten durch etwaige Unterschiede zwischen partieller und totaler Übereinstimmung. Mit den Methoden von [Mo 74a] oder [SZ 73] läßt sich diese Konstruktionsidee dann auf eine PRAS mit nur einem Speicher übertragen (Anordnung von Arbeits- und Assoziativbereich hintereinander in einem

Speicher, Rechnung mit $2.z$ statt mit z , Verwendung jedes zweiten Registers zur Adressenverkettung zwischen Arbeits- und Assoziativbereich). Die genaueren Einzelheiten sollen

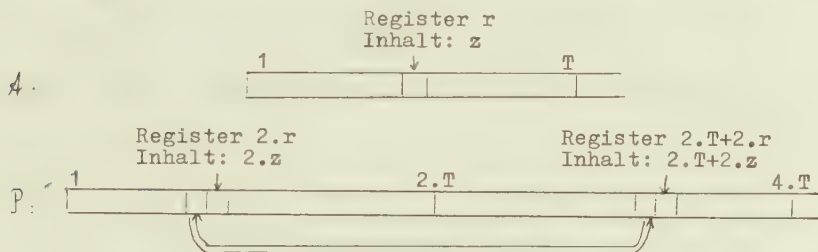


Abb.24: Simulation einer RAS $\mathcal{A}$ durch eine PRAS $\mathcal{P}$

hier nicht ausgeführt werden, da es sich hier um übliche Techniken für solche Maschinenmodelle handelt; sie sind in Abb. 24 angedeutet. Der Schwerpunkt dieses Abschnitts sollte mehr bei den allgemeinen Sätzen über die Simulation von Random-Access-Maschinen mit assoziativem Speicherzugriff durch solche ohne einen assoziativen Zugriff liegen.

Dabei zeigt Satz 7.5, daß der Gewinn an Effizienz durch das Hinzufügen assoziativer Zugriffsooperationen zu den normalen Operationen einer RAM nicht allzu groß ist - wobei allerdings noch einmal darauf hingewiesen werden soll, daß hier nur von assoziativem Zugriff und nicht von assoziativer (Parallel-)Verarbeitung die Rede ist. Satz 7.5 stellt gegenüber dem bisherigen Ergebnis aus [Ha 71b] (Satz 7.1) in gewisser Weise eine Verbesserung dar, wobei freilich zu beachten ist, daß bei dem hier verwendeten Maschinenmodell die Einschränkung in den arithmetischen Operationen - die für das Modell von [Ha 71b] nicht gilt -

wesentlich verwendet wurde; durch sie ist gewährleistet, daß die von der Maschine berechneten Zahlen nicht allzu groß werden. Andererseits zeigt Satz 7.4 aber auch, daß das Einbeziehen von Präfixen der Registerinhalte als Schlüssel für einen assoziativen Zugriff keine stärkeren arithmetischen Operationen der simulierenden RAS bzw. RAM notwendig macht. Ebenso wie Satz 6.3 lassen sich schließlich auch Satz 7.2 und 7.5 auf Maschinenmodelle mit allgemeineren arithmetischen Operationen (allgemeine Addition oder Multiplikation) übertragen, wenn dann zusätzlich als Bedingung vorausgesetzt wird, daß die im Verlauf von $T(n)$ Schritten berechneten Registerinhalte nicht größer als $(T(n))^C$ für eine feste konstante C werden. Es bleibt noch offen, ob ein zu Satz 7.2 und 7.5 analoges Ergebnis auch dann gilt, wenn diese zusätzliche Bedingung wegfällt. Zwar können dann in $T(n)$ Schritten Zahlen bis zur Größenordnung $T(n)$ berechnet und dementsprechend auch Adressen bis zu dieser Größenordnung benutzt werden, doch ist die Anzahl der benutzten Register auch hier durch $T(n)$ beschränkt. Ähnlich wie im Beweis zu Satz 7.2 in Adressenspeicher die Datenstruktur eines binären Baumes aufgebaut wurde, könnte man bei der allgemeinen RAM durch einen im Verlauf der Rechnung aufzubauenden sogenannten ausgeglichenen Baum ("balanced tree") eine Buchführung über die benutzten Adressen und damit auch über die Speicherbereiche zwischen ihnen speichern. Zu jedem Unterbaum müßte dann wieder im Markenspeicher Information über die Anzahl des Auftretens einer Marke in dem entsprechenden Speicherbereich

vorhanden sein. Während bei Satz 7.2 aber diese Speicherbereiche fest bleiben, müßten sie hier bei einem neuen Ausbalancieren des Baumes jeweils neu angepaßt werden, und es ist nicht zu sehen, wie hier für alle "Marken" in kürzerer Zeit als in $T(n)$ Schritten pro Schritt der RAS die Buchführung im Markenspeicher auf den neuen Stand gebracht werden könnte.

Überlegt man noch einmal, worin letztlich - für das hier verwendete Maschinenmodell - die Verbesserung von Satz 7.2 und 7.5 gegenüber Satz 7.1 begründet ist, so findet man als Grundgedanken, daß die im Beweis zu Satz 7.1 verwendete sequentielle Suche hier ersetzt wird durch eine binäre Suche. Als zweiter wesentlicher Schritt tritt bei Satz 7.2 hinzu, daß sich eine zweidimensionale RAM mit Zeitbeschränkung $T(n)$ durch eine eindimensionale RAM mit Zeitbeschränkung $T(n) \cdot \log T(n)$ (statt $T(n)^2$, wie bei einer naiven Methode) simulieren läßt. Eine genauere Betrachtung des Beweises in [Fl 74] zeigt, daß auch diese Simulation ihre Effizienz durch das Prinzip der binären Suche gewinnt.

Während die Sätze 7.2 und 7.5 eine Aussage darüber machen, wieviel effizienter eine RAS gegenüber einer RAM höchstens ist, sind nach wie vor Aussagen darüber unbekannt, ob es überhaupt Fälle gibt, für die sich eine Überlegenheit der RAS gegenüber der RAM bei gleicher Zeitbeschränkung (von Konstanten abgesehen) beweisen läßt. Ein solcher Beweis bräuchte Methoden, um für Modelle wie die RAM nichttriviale untere Schranken für die Komplexität von Berechnungen zu finden, die bisher noch nicht bekannt sind.

Einzigste Ergebnisse in dieser Richtung bleiben weiter die Ergebnisse aus Abschnitt 3, die zeigen, daß für das Modell der Turingmaschine und im speziellen Fall der On-Line-Spracherkennung zusätzliche assoziative Operationen einen beweisbaren Zeitgewinn bringen können.

Literatur

- [CA 69] COOK, Stephen A., und AANDERAA, Stål O.: On the Minimum Computation Time of Functions
Transactions of the American Mathematical Society 142 (1969), 291 - 314
- [CR 73] COOK, Stephen A., und RECKHOW, Robert A.: Time Bounded Random Access Machines
Journal of Computer and System Sciences 7 (1973), 354 - 375
- [Fl 74] FLEISCHHAUER, Günther: Beziehungen zwischen den Zeitkomplexitätsklassen verschiedener Automatenmodelle
Bericht Nr. 10, Institut für Informatik, Universität Hamburg, Sept. 1974
- [FR 68] FISCHER, Michael J., und ROSENBERG, Arnold L.: Limited Random Access Turing Machines
IEEE Conference Record of 1968 Ninth Annual Symposium on Switching and Automata Theory, 356 - 367
- [Gr 74] GREIBACH, Sheila A.: Jump PDA's and Hierarchies of Deterministic Context-Free Languages
SIAM Journal on Computing 3 (1974), 111 - 127
- [Je 65] JESSEN, Eike: Assoziative Speicherung
Elektronische Datenverarbeitung, Beiheft 5, 1965
- [Ha 71a] HARTMANIS, Juris: Size Arguments in the Study of Computation Speeds
Symposium on Computers and Automata, Microwave Research Institute Symposia Series vol. XXI, Polytechnic Press, Brooklyn (N.Y.) 1971, 7 - 18
- [Ha 71b] HARTMANIS, Juris: Computational Complexity of Random Access Stored Program Machines
Mathematical Systems Theory 5 (1971), 232 - 245

- [He 66] HENNIE, C.F.: On-Line Turing Machine Computations
IEEE Transactions on Electronic Computers
EC - 15 (1966), 35 - 44
- [HS 65] HARTMANIS, Juris, und STEARNS, R.E.: On the Computational Complexity of Algorithms
Transactions of the American Mathematical Society 117 (1965), 285 - 306
- [HS 74] HARTMANIS, Juris, und SIMON, J.: On the Power of Multiplication in Random Access Machines
Technical Report 74-213, Department of Computer Science, Cornell University, Sept. 1974
- [HU 69] HOPCROFT, John E., und ULLMANN, Jeffrey D.: Formal Languages and their Relation to Automata
Addison-Wesley, Reading (Mass.) 1969
- [Mo 74a] MONIEN, Burkhard: Komplexitätsklassen von Automatenmodellen und beschränkte Rekursion
Bericht Nr. 8, Institut für Informatik, Universität Hamburg, Febr. 1974
- [Mo 74b] MONIEN, Burkhard: Characterization of Time-Bounded Computations by Limited Primitive Recursion
Second Colloquium on Automata, Languages and Programming, Saarbrücken 1974, 280 - 293
- [Sch 69] SCHÖNHAGE, A.: Universelle Turing Speicherung
Automatentheorie und Formale Sprachen (Hrsg. J.Dörr, G.Hotz), Tagungsbericht Oberwolfach 1969, 369 - 383
- [Sch 74] SCHNORR, C.P.: Rekursive Funktionen und ihre Komplexität
Teubner Verlag, Stuttgart 1974
- [Sei 74] SEIFERAS, Joel J.: Nondeterministic Time and Space Complexity Classes
MIT, Project MAC, MAC-TR-137, Sept. 1974
- [St 71] STOSS, H.-J.: Zwei-Band Simulation von Turingmaschinen
Computing 7 (1971), 222 - 235

- [SZ 73] SUDBOROUGH, J.H., und ZALCBERG, A.: On Families of Languages Defined by Time-Limited Random Access Machines
Mathematical Foundations of Computer Science, Proceedings of Symposium and Summer School High Tatras, 1973, 333 - 338
- [We 70] WEICKER, Reinhold: Erkennung kontextfreier Sprachen durch Tabulator-Turingmaschinen
Arbeitsberichte des Instituts für Mathematische Maschinen und Datenverarbeitung, Universität Erlangen-Nürnberg, Bd. 3 Nr. 5, Juli 1970
- [We 71] WEICKER, Reinhold: Tabulator-Turingmaschine und Komplexität
Computing 7 (1971), 264 - 274, und 9 (1972), 165 - 167
- [We 74] WEICKER, Reinhold: Turing Machines with Associative Memory Access
Second Colloquium on Automata, Languages and Programming, Saarbrücken 1974, 458 - 472
- [WM 73] WEICKER, Reinhold, und MONIEN, Burkhard: Über die Effizienz von Tabulator-Turingmaschinen mit beschränkter Markenzahl im Vergleich zu anderen Maschinenmodellen
Bericht Nr. 6, Institut für Informatik, Universität Hamburg, Okt. 1973

L e b e n s l a u f

Am 2. Dezember 1944 wurde ich als drittes von vier Kindern des Dr.-Ing. Reinhold Weicker und seiner Ehefrau Margarete, geb. Bornhäuser, in Geislingen/Steige in Württemberg geboren. Nach Kriegsende lebten wir bis 1949 in Freiburg/Breisgau, dann in Hof/Saale, ab 1953 in Erlangen. In Hof und Erlangen besuchte ich die Volksschule (1950 - 1954) und dann das humanistische Gymnasium Fridericianum Erlangen (1954 - 1963). Im Wintersemester 1963/64 begann ich, an der Universität Erlangen-Nürnberg Mathematik zu studieren. Nach der Diplom-Vorprüfung (WS 1965/66) studierte ich zwei Semester in Göttingen, dann ab SS 1967 wieder in Erlangen, nun mit dem Wahlfach Datenverarbeitung an der neu entstandenen Technischen Fakultät. Meine Diplomarbeit schrieb ich im Gebiet der theoretischen Informatik, im Oktober 1970 legte ich die Diplomprüfung ab. Seitdem bin ich im Gebiet der Informatik als wissenschaftlicher Assistent tätig, zunächst an der Technischen Universität Berlin (Forschungsgruppe Informationsverarbeitung II, Prof. Dr. H.-J. Schneider), dann ab August 1972 an der Universität Hamburg (Forschungsgruppe Automatentheorie und Formale Sprachen, Prof. Dr. W. Brauer).

Offsetdruck-Fotodruck

J. Hogl, 8520 Erlangen, Hauptstr. 109 (Nähe Martin-Luther-Platz) Tel. 22990
